Министерство образования Республики Беларусь

Учреждение образования
«Полоцкий государственный университет»

КОМПЬЮТЕРНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Учебно-методический комплекс
для студентов экономических специальностей

В трех частях

Часть 2

С. Е. Рясова
Д. А. Оськин

Основы программирования
на Visual Basic For Applications в Microsoft Excel.
Сетевые информационные технологии

Новополоцк
ПГУ
2016

УДК 004.3(075.8)
ББК 32.97я73
К63

Рекомендовано к изданию
учебно-методической комиссией финансово-экономического факультета
в качестве учебно-методического комплекса
(протокол № 6 от 30.06.2014)

РЕЦЕНЗЕНТЫ:

заместитель начальника отдела АСУ
(по программированию и эксплуатации программного обеспечения)
завода «Полимир» ОАО «Нафтан» И. Н. САС;
старший преподаватель кафедры
технологии и методики преподавания УО «ПГУ»
С. А. ТОЛОКНОВА

К63 **Компьютерные информационные технологии** : учеб.-метод. комплекс для студентов экон. специальностей. В 3 ч. Ч. 2. Основы программирования на Visual Basic For Applications в Microsoft Excel. Сетевые информационные технологии / С. Е. Рясова, Д. А. Оськин. – Новополоцк : ПГУ, 2016. – 320 с.
ISBN 978-985-531-530-9.

В 2012 году вышла 1 часть в 2-х книгах учебно-методического комплекса автора С.Е. Рясовой.

Приведены темы изучаемого курса, объем в часах лекционных и лабораторных занятий, вопросы к экзамену, примеры тестовых заданий.

Предназначен для студентов и преподавателей экономических специальностей вуза.

УДК 004.3(075.8)
ББК 32.97я73

ISBN 978-985-531-530-9 (ч. 2)
ISBN 978-985-531-312-1

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В соответствии с учебной программой дисциплины «Компьютерные информационные технологии» для экономических специальностей настоящий учебно-методический комплекс включает в себя два раздела: «Основы программирования на Visual Basic for Applications в Microsoft Excel» и «Сетевые информационные технологии».

В разделе «Основы программирования на Visual Basic for Applications в Microsoft Excel» рассмотрены вопросы решения задач с использованием элементов программирования на Excel VBA, в том числе пользовательский интерфейс редактора Excel VBA; структура языка Excel VBA; простейшие операторы и конструкции Excel VBA; основные сведения о процедурах и функциях. Особое внимание уделено таким темам, как «Обмен данными с рабочей книгой MS Excel», «Проектирование диалоговых окон».

В разделе «Сетевые информационные технологии» рассмотрены вопросы классификации и стандартизации компьютерных сетей; история развития и основы функционирования глобальной сети Интернет, а также основы языка гипертекстовой разметки HTML.

ЛЕКЦИОННЫЙ КУРС

I. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА VISUAL BASIC FOR APPLICATIONS В MICROSOFT EXCEL

1. Технологии и инструментальные средства программирования

Язык программирования – это набор символов и терминов, который в соответствии с правилами синтаксиса описывает алгоритм решения задачи.

Программа на языке программирования представляет собой совокупность операторов, записанных в соответствии с принятым синтаксисом.

Программирование (programming) – это процесс создания последовательности действий (операций), проводимый в целях достижения требуемого результата.

Программирование, осуществляемое программистом, непосредственно связано с языком и инструментальным программным обеспечением, поддерживающим разработку. В состав данного программного обеспечения входят: транслятор и база данных с набором стандартных программ. Процесс программирования состоит из следующих стадий:

- формулирование задачи;
- разработка программы, включая кодирование и тестирование;
- создание новых версий.

Существуют сотни реально использующихся языков программирования, каждый для своей области применения. Уровень языка программирования определяется в зависимости от степени детализации алгоритма. Причем, чем меньше детализация, тем выше уровень языка. Различают три уровня языков программирования:

- машинные;
- машинно-ориентированные (*ассемблеры* – языки программирования, близкие к программированию непосредственно в машинных кодах. Они, как правило, используют особенности конкретного семейства процессоров);
- машинно-независимые (языки высокого уровня).

Машинные и машинно-ориентированные языки – это языки *низкого уровня*, которые требуют указания мелких деталей процесса обработки данных.

Языки высокого уровня во многом имитируют естественные, используют многие разговорные слова, общепринятые математические символы. Различают языки следующих групп: алгоритмические, предназначенные для однозначного описания алгоритмов (такие как *Basic*, *Pascal*,

C), логические – ориентированные не на разработку алгоритма решения задачи, а на систематическое и формализованное описание задачи (например, *Prolog, Lisp*), объектно-ориентированные, основанные на понятии объекта и действиях над ним (к примеру, *Object Pascal, C++, Java*).

Языки программирования подробно изучаются при подготовке профессиональных программистов.

2. Объектно-ориентированное программирование

Объектно-ориентированное программирование (ООП) представляет собой новый этап развития современных концепций построения языков программирования.

В основе ООП лежит понятие объекта (`Object`), сочетающего в себе данные и действия над ними – методы (`Methods`).

ООП характеризуется тремя основными свойствами: *инкапсуляция*, *наследование*, *полиморфизм*.

Наследование позволяет создавать иерархию объектов, начиная с некоторого простого первоначального (*предка*) и заканчивая более сложными, но включающими (*наследующими*) свойства предшествующих элементов (*потомки*). Эта иерархия в общем случае может иметь довольно сложную древовидную структуру. Каждый потомок несет в себе характеристики своего предка (содержит те же *данные* и *методы*), а также обладает собственными характеристиками. При этом наследуемые данные и методы описывать у потомка нет необходимости.

Задаваемый объект позволяет локализовать в одном месте его свойства и сделать его в некотором смысле замкнутым по отношению к другим объектам и элементам программы, что, конечно, в ряде случаев позволяет упростить его программирование.

Пример. 2.1. В автомобиле есть коробка передач, есть система управления и т. д. Эти элементы взаимодействуют с внутренними элементами, но для того чтобы ехать не обязательно знать, как все это работает, а нужно нажать на педали, включить передачу и т. д. – принцип функционирования автомобиля скрыт от водителя. Это и есть ИНКАПСУЛЯЦИЯ (скрытие реализации).

Полиморфизм подразумевает, что различные объекты могут по-разному реагировать на одинаковые внешние события в зависимости от того, как реализованы их методы.

ООП обладает рядом преимуществ при создании больших программ. В частности, к ним можно отнести:

– использование более естественных с точки зрения повседневной практики понятий, простота введения новых;

– некоторое сокращение размера программ за счет того, что повторяющиеся (наследуемые) свойства и действия можно не описывать многократно;

– использование динамических объектов позволяет более эффективно использовать оперативную память;

– возможность создания библиотеки объектов;

– возможность написания подпрограмм с различными наборами формальных параметров, но имеющих одно и то же имя;

– возможность разделения доступа к различным объектам программы и т. д.

Однако следует иметь в виду, что ООП обладает и рядом недостатков и эффективно не во всех случаях. Неэффективно ООП применительно к небольшим программам, поэтому его можно рекомендовать при создании больших программ, а лучше даже класса программ. Можно сказать, что ООП упрощает скорее не саму программу, а упрощает технологию ее создания.

3. Что такое VBA?

По инициативе главы корпорации Microsoft Билла Гейтса (Bill Gates) простой и легкий в освоении язык программирования высокого уровня *Basic* лег в основу мощной среды программирования для широкого круга приложений, выпускаемых как самостоятельные продукты этой фирмы.

VBA (*Visual Basic for Applications*) – представляет собой подмножество широко распространенного в настоящее время языка *Visual Basic*. Язык специально ориентирован на работу в среде Microsoft Office и создавался как средство записи макросов.

Впервые он появился в Microsoft Excel 5.0, а затем и другие приложения Microsoft Office перешли на его использование.

Визуально концепцию VBA можно отобразить как на рисунке I.3.1.

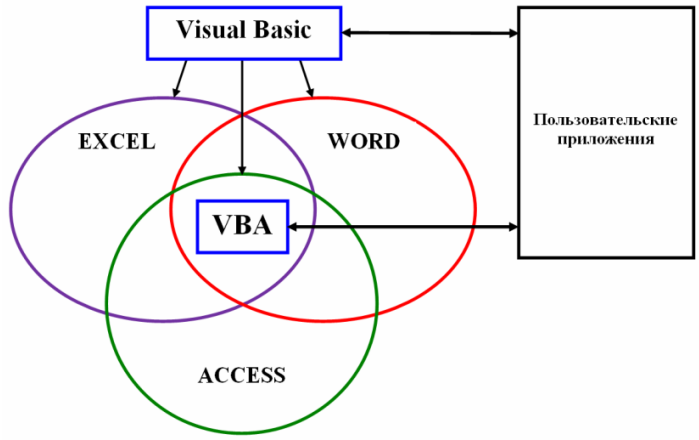


Рис. I.3.1. Визуальная концепция VBA

Таким образом, VBA находится в «центре» всех приложений, объединяя их методологически в аспектах проектирования прикладных приложений.

4. Принципы разработки приложений электронных таблиц

Разработчики электронных таблиц несут ответственность за выполнение следующих требований:

– определение потребностей пользователя;

– планирование приложения, которое соответствует этим условиям;

– разработку наиболее подходящего интерфейса пользователя;

– создание электронной таблицы, формул, макросов и пользовательского интерфейса;

– тестирование приложения в разных условиях;

– изменение приложения с целью повышения его надежности и отказоустойчивости (часто по результатам тестирования);

– эстетическую привлекательность и наглядность приложения;

– документирование усилий, потраченных на разработку;

– размещение приложения в компьютере пользователя;

– обновление приложения в случае необходимости.

Перечисленные требования не являются обязательными для выполнения при создании каждого приложения. Кроме того, порядок их выполнения в разных проектах может быть разным.

4.1. Определение потребностей пользователя

Когда разработчик приступает к разработке проекта приложения электронных таблиц, то одним из первых действий является точное определение потребностей конечного пользователя.

Ниже приведены основные правила, придерживаясь которых можно облегчить начальную фазу разработки:

– если существует возможность, то необходимо говорить непосредственно с потенциальными пользователями приложения, а не только с руководителем проекта;

– следует узнать, что уже сделано (если только сделано) для удовлетворения потребностей пользователей. Возможно, удастся сэкономить часть времени, переделав уже существующее приложение. В крайнем случае, изучая уже имеющиеся решения, можно подробно ознакомиться с работой конечных пользователей;

– определить, какие ресурсы имеются в распоряжении пользователя (например, узнать, существуют ли какие-либо аппаратные или программные ограничения, которые нужно учитывать);

– узнать, какая версия (или версии) Microsoft Excel используется в системе;

– узнать уровень квалификации конечных пользователей;

– определить, как долго будет использоваться приложение и ожидаются ли изменения в нем в будущем. Знание этого вопроса окажет влияние на выполняемые операции и поможет спланировать изменения.

4.2. Проектирование приложения, соответствующего потребностям пользователей

Определив потребности конечных пользователей, можно приступать к проектированию приложения. На начальном этапе планирования проекта необходимо продумать следующие вопросы:

– *файловая структура*. Здесь следует определить, будет ли использоваться одна рабочая книга с множеством листов или же несколько однолистных рабочих книг или файл шаблона;

– *структура данных*. Необходимо учесть структуру данных, которые будут использоваться в приложении. В том числе необходимо решить, использовать файлы внешних баз данных или хранить всю информацию в рабочих листах;

– *формулы или VBA*. Следует определить, что требуется для проведения вычислений: использование формул или написание процедур VBA? Каждый из представленных вариантов имеет свои достоинства и недостатки;

– *версия Excel*, где будет запускаться Excel-приложение;

– *как обрабатывать ошибки*. Для приложений немаловажным вопросом является обработка ошибок. Следует определить, как приложение будет «отлавливать» ошибки, и что потом оно будет с ними делать. Например, если приложение применяет форматирование к активному рабочему листу, то необходимо предусмотреть случай, когда активным будет лист диаграмм.

– *использование специальных возможностей*. Если в приложении будет суммироваться большое количество данных, то нужно подумать над использованием такого средства Excel, как сводные таблицы. Также потребуется такая возможность Excel, как проверка данных, чтобы тестировать вводимые данные на правильность;

– *вопросы производительности*. Решить вопрос увеличения производительности и эффективности приложения следует еще на стадии проектирования;

– *уровень безопасности*. В Excel предусмотрено несколько вариантов защиты, которые призваны предотвратить доступ к определенным элементам рабочей книги. Например, можно блокировать ячейки, чтобы нельзя было изменить находящиеся в них формулы, или назначить пароль, чтобы неавторизованные пользователи не смогли просматривать определенные файлы или получать доступ к ним.

4.3. Определение удобного пользовательского интерфейса

Разрабатывая электронные таблицы, которые будут использоваться другими людьми, нужно обратить особое внимание на пользовательский интерфейс. *Пользовательский интерфейс* – это метод взаимодействия пользователя с приложением – щелчки на кнопках, использование меню, нажатие клавиш, доступ к панелям инструментов и т. д.

В Excel содержится несколько средств, необходимых при разработке пользовательского интерфейса:

– пользовательские диалоговые окна (пользовательские формы);

– элементы управления (такие, например, как *ListBox* (поле со списком) и *CommandButton* (командная кнопка)), размещаемые непосредственно на рабочем листе;

– пользовательские меню;

– пользовательские панели инструментов;

– пользовательские сочетания клавиш.

4.3.1. Создание пользовательских диалоговых окон

Пользовательские диалоговые окна играют важную роль в тех интерфейсах, которые разрабатываются для приложений.

С помощью пользовательского диалогового окна пользователь может ввести те или иные данные, получить выбранные им варианты или предпочтения, а также задать ход выполнения всего приложения. Такие диалоговые окна хранятся в пользовательских формах (*UserForm*). Элементы, из которых состоит диалоговое окно, называются элементами управления (к ним относятся кнопки, раскрывающиеся списки, флажки и т.д.). Они также называются *элементами управления ActiveX*.

4.3.2. Использование элементов управления ActiveX в рабочем листе

В Excel элементы управления ActiveX, предназначенные для пользовательских форм, могут встраиваться непосредственно на рабочий лист. На рисунке I.4.1. представлена простая модель рабочего листа с несколькими элементами управления. На рабочем листе находятся переключатели, кнопки, поле, а также поле со списком.

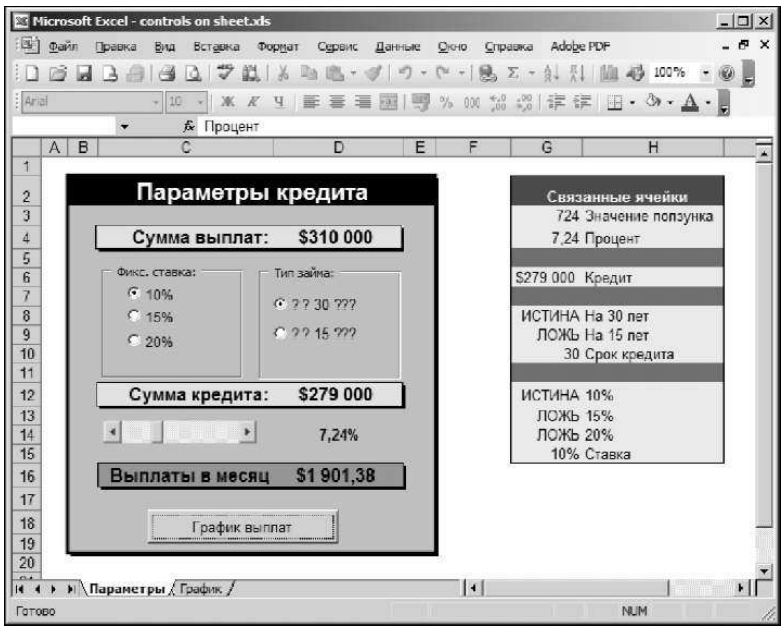


Рис. I.4.1. Элементы управления, размещенные на рабочем листе

Использование элементов управления непосредственно на рабочем листе отменяет необходимость в создании пользовательских диалоговых окон. Часто бывает так, что вставка в рабочий лист нескольких элементов управления ActiveX значительно упрощает работу с электронной таблицей. В результате пользователь сможет выбирать то, что ему нужно, работая со знакомыми элементами управления, а не вводя значения в ячейки.

Элементы управления ActiveX отображаются на панели инструментов «Элементы управления» (рис. I.4.2).

Рис. I.4.2. Панель инструментов «Элементы управления»

Более подробно об элементах управления ActiveX см. раздел 6.3.2 «Работа с пользовательскими формами».

4.3.3. Настройка меню

В приложениях электронных таблиц управлять пользовательским интерфейсом можно самыми разными способами. Речь идет об изменении стандартных меню Excel или создании собственных систем меню. Вместо того чтобы создавать кнопки для выполнения макросов, можно добавить одно или несколько меню (а также элементов меню) и уже с их помощью запустить созданные ранее макросы. Преимущество пользовательских меню заключается в том, что строка меню всегда отображается на экране, а кнопка, размещенная на рабочем листе, при прокрутке может быстро исчезнуть из поля зрения.

Существует два способа настройки меню Excel. Изменения в меню можно внести с помощью кода VBA или посредством редактирования самих меню (для этого нужно воспользоваться командой *меню Вид → Панели инструментов → Настройка*).

Настраивать можно любые меню, отображаемые программой Excel, даже контекстные, которые появляются при щелчке на объекте правой кнопкой мыши. На рисунке I.4.3 показано пользовательское контекстное меню, которое появляется при щелчке правой кнопкой мыши на ячейке или диапазоне ячеек. Обратите внимание, что в этом меню находятся только новые команды, которые по умолчанию не используются.

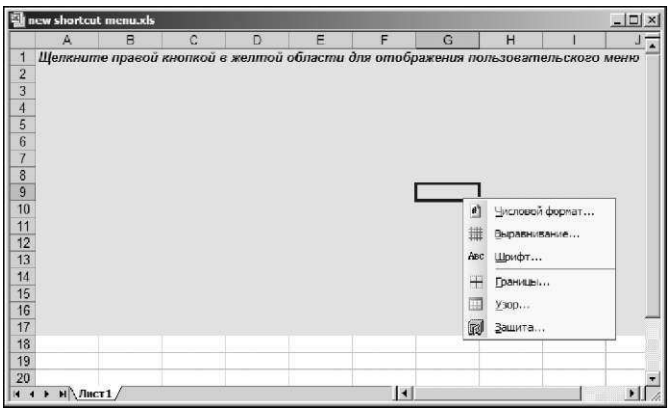


Рис. I.4.3. Пример пользовательского контекстного меню

4.3.4. Настройка панелей инструментов

Панели инструментов широко распространены в приложениях Windows, и в программу Excel встроено огромное их количество. Обычно кнопки панелей инструментов помогают пользователям быстрее выполнять наиболее распространенные команды меню. Так как для щелчка на кнопке, которая расположена на панели инструментов, обязательно требуется мышь, то такая кнопка – далеко не единственное средство выполнения

конкретной операции. Однако в Excel большинство самых распространенных операций в электронных таблицах выполняется без использования меню, т. е. для их вызова вполне хватает панелей инструментов.

Можно создать пользовательскую панель инструментов, на которой будут содержаться кнопки только тех команд, которые необходимо сделать доступными для пользователей. Если с этими кнопкам связаны макросы, то пользовательская панель инструментов станет эквивалентной группе кнопок, расположенных на рабочем листе. Однако применять панель инструментов удобнее, поскольку она всегда отображается на экране, и ее перемещение в другое место не повлияет на основной документ. Что же касается кнопок, размещенных на рабочем листе, то они жестко прикреплены к определенному месту, и их можно убрать из поля зрения в результате прокрутки документа.

Приложение можно настроить таким образом, что при его загрузке на экране будет появляться соответствующая панель инструментов.

4.3.5. Создание сочетаний клавиш

Excel позволяет назначать макросу сочетание клавиш, в состав которого входит Ctrl (или Shift+Ctrl). Когда пользователь нажимает сочетание клавиш, макрос запускается и выполняет все определенные в нем действия. Если существенную роль в управлении приложением играет скорость выполнения операций, то сочетания клавиш использовать эффективнее, поскольку на их нажатие обычно уходит меньше времени, чем на выбор команд меню, применение панели инструментов или опций диалоговых окон.

4.3.6. Разработка приложения

Когда определены потребности пользователя, подобран тип проекта и решено, какие компоненты необходимо применить в пользовательском интерфейсе, настает время завершить подготовительные операции и приступить к созданию самого приложения. Этот этап, конечно же, занимает значительную часть времени, которое тратится на реализацию проекта.

Каким образом будет создаваться приложение, зависит от персонального стиля работы разработчика и от природы самого приложения.

4.4. Работа с конечным пользователем

В настоящем разделе речь пойдет о весьма важных проблемах разработки, которые дают о себе знать, когда приложение становится работоспособным и приближается время размещать готовый проект в компьютерах конечных пользователей.

4.4.1. Тестирование приложения

Создав приложение, его необходимо протестировать. Это один из самых важных этапов. Нередко бывает так, что на тестирование и отладку приложения уходит столько же времени, сколько тратится на создание его исходного варианта.

Склонность к ошибкам имеют не только стандартные компилированные приложения. То же самое можно сказать и о приложениях электронных таблиц. Ошибка обычно определяется следующим образом:

1) это нечто такое, что при выполнении программы (или приложения) происходит, но происходить не должно;

2) это нечто такое, что не происходит, но происходить как раз обязано.

Значительную часть времени, которое уйдет на разработку, нужно заранее выделить на тестирование приложения во всех возможных условиях и на исправление любых встретившихся проблем.

Вероятно, не требуется напоминать об обязательности тщательного тестирования любых электронных таблиц, которые разрабатываются для других. Учитывая возможную аудиторию приложения, можно сделать его отказоустойчивым. Другими словами, надо попытаться прогнозировать все возможные ошибки и недочеты, которые потенциально могут допускаться пользователями, и приложить усилия, чтобы их избежать.

4.4.2. Как сделать приложение отказоустойчивым

Нарушить электронную таблицу можно очень легко. Часто бывает так, что удаление одной важной формулы или ключевого значения вызывает ошибки во всей таблице или даже в других зависимых таблицах. Более того, если поврежденную таблицу сохранить, то на диске она заменит корректную копию.

В Excel для защиты рабочих листов и их частей можно использовать несколько способов:

– заблокировать определенные ячейки, чтобы предотвратить их изменение (задается с помощью вкладки «Защита» диалогового окна «Формат ячеек»). Блокировка вступает в силу только тогда, когда с помощью команды *меню Сервис → Защита → Защитить лист* документ защищается паролем;

– добавить защиту всей рабочей книги – ее структуры, расположения и размеров окна или одновременно всех перечисленных выше параметров. Это можно сделать с помощью команды *меню Сервис → Защита → Защитить книгу*;

– скрыть формулы, расположенные в ячейках, чтобы их не смогли увидеть другие пользователи (задается с помощью вкладки «Защита» диалогового окна «Формат ячеек»). Формула будет скрыта только тогда, когда с помощью команды *меню Сервис → Защита → Защитить лист* документ будет защищен паролем;

– заблокировать объекты рабочего листа задается с помощью вкладки «Защита» диалогового окна «Формат ячеек»). Эта блокировка вступает в силу только тогда, когда с помощью команды *меню Сервис → Защита → Защитить лист* документ защищается паролем;

– скрыть строки (команда *меню Формат → Строка → Скрыть*), столбцы (команда *меню Формат → Столбец → Скрыть*), листы (команда *меню Формат → Лист → Скрыть*) и документы (команда *меню Окно → → Скрыть*). В результате этих операций лист не будет выглядеть загроможденным, однако он частично защищен;

– превратить рабочую книгу Excel в документ только для чтения (и защищенную паролем), чтобы ее файл нельзя было изменить. Этот режим устанавливается в диалоговом окне «Параметры сохранения». Его можно запустить из диалогового окна «Сохранение документа» по команде *меню Сервис → Общие параметры*.

– добавить пароль, чтобы неавторизованный пользователь не мог открыть файл. Такой режим можно установить в диалоговом окне «Параметры сохранения». Чтобы его отобразить, нужно выбрать в диалоговом окне «Сохранение документа» команду *меню Сервис → Общие параметры*.

– использовать защищенные паролем надстройки, чтобы пользователь не мог изменить рабочие листы самой надстройки.

4.4.3. Привлекательное и наглядное приложение

Разрабатывая электронные таблицы для других пользователей, необходимо уделять особое внимание внешнему виду приложения.

У конечных пользователей первая реакция на приложение вызвана исключительно внешним видом программы. Это в полной мере относится и к приложениям, которые разрабатываются с помощью Excel. Хороший внешний вид приложения говорит о том, что разработчик настолько заботился о своем продукте, что не пожалел времени и усилий на создание всесторонне качественного интерфейса. Чтобы создать качественный интерфейс приложения, следует руководствоваться следующими принципами:

– добиваться единообразия. Разрабатывая, например, диалоговые окна, стараться по возможности следовать внешнему виду диалоговых окон Excel. Соблюдать единообразие в формате, шрифтах, размере текста и цветах;

– распространенной ошибкой разработчиков является то, что они пытаются вывести на экран или в диалоговом окне как можно больше информации. Добиваясь этого, надо следовать эмпирическому правилу, согласно которому информацию необходимо выдавать порциями – не более одного-двух блоков за раз;

– если для получения информации от пользователя используется диалоговое окно, то имеет смысл разбить его на несколько окон, каждое из которых будет менее загроможденным. Если же решено использовать сложное диалоговое окно, то чтобы его разбить, можно воспользоваться элементом управления *MultiPage* (*Множество вкладок*), который помогает создать диалоговое окно с несколькими вкладками;

– осторожно использовать цвета;

– уделить особое внимание числовым форматам, начертанию, размерам шрифтов, а также границам.

Эстетика – понятие довольно субъективное, но если вы сомневаетесь в своих способностях, то постарайтесь добиться как минимум простоты и ясности интерфейса приложения.

4.4.4. Создание пользовательской справочной системы

В пользовательской документации используется преимущественно два варианта: документация в бумажном и электронном виде.

Электронная справочная система – это стандартный компонент Windows-приложений. Excel-приложения не исключения – они могут иметь электронную справочную систему. На разработку электронной справочной системы требуется много времени и усилий.

4.4.5. Документирование процесса разработки

В процессе создания приложения важно тщательно документировать свою работу. Созданная документация пригодится тогда, когда потребуется вернуться к доработке приложения. Она также понадобится тому, кому в дальнейшем будет передано приложение.

Как же документировать приложение электронных таблиц? Информацию можно хранить в рабочей книге или другом файле. Можно представить документацию в виде бумажного документа. Возможно, самый легкий способ – это использовать отдельный рабочий лист, чтобы хранить в нем комментарии и основную информацию о проекте. Что касается кода VBA, то в нем можно использовать комментарии.

4.4.6. Распространение приложения среди пользователей

Существуют следующие способы распространения приложения среди пользователей:

– создать диск с приложением и внести в него простые инструкции по использованию;

– создать официальную программу установки. Эта программа предназначена для автоматического выполнения всех необходимых операций по внедрению приложения в компьютер пользователя. Такую программу можно написать на традиционном языке программирования, купить общий ее код или написать свою собственную на VBA.

4.4.7. Обновление приложения

Распространением приложения среди пользователей завершается процесс его разработки. Однако случается так, что пользователи приложения не полностью им довольны. Кроме того у пользователя может возникнуть необходимость в дополнительных функциях, поэтому он попросит об их включении в приложение, т. е. потребуется выполнить *обновление приложения*.

5. Visual Basic for Applications в Microsoft Excel

На сегодняшний день Microsoft Excel является самой востребованной программой для работы с числовыми данными. Если взглянуть на типичные выполняемые действия с рабочими книгами Microsoft Excel, то это, прежде всего, оформление данных на листах, размещение формул, построение диаграмм и т. п. (эти возможности были рассмотрены в части 1 настоящего учебно-методического комплекса). Однако это составляет только часть ресурсов, которые заложены в рассматриваемое приложение. Следует отметить, что в реальных проектах автоматизации офисной деятельности часто требуется разрабатывать сложные интерфейсы ввода информации, реализовывать нетривиальные алгоритмы обработки данных и формировать сложные отчеты. В этом случае представление об объектах Microsoft Excel и знание языка программирования Visual Basic for Applications (далее VBA) открывает широкие возможности для создания своих собственных прикладных офисных решений. На языке VBA создаются *модули* – объекты, содержащие созданные пользователем процедуры. *Процедура* представляет собой совокупность операторов языка VBA, реализующую ряд логических шагов для выполнения конкретного действия.

5.1. Объекты Microsoft Excel. Свойства, методы и события объектов в Excel VBA

VBA является объектно-ориентированной средой, содержащей большой набор объектов, каждый из которых обладает множеством свойств и методов.

Сам объект и все, что нужно для выполнения определенных действий с ним (создание, отображение и т.п.), содержится в определенном классе. Имя класса указывает на тип объектов, которые он содержит. Классы можно рассматривать как основу для создания других объектов этого же типа. Поэтому все объекты одного класса, например класса *TextBox*, будут «действовать» одинаково.

Свойства и методы являются членами класса. С помощью свойств описывается, как выглядит объект, в том числе дается информация о форматировании текста, цвете, размере объекта и т.д. Методы являются процедурами (множеством операторов, осуществляющих определенную задачу), которые могут быть выполнены для объекта (процедуры создания и удаления объекта, процедуры событий, определяющие взаимодействие объекта с пользователем, и т.д.)

5.1.1. Объекты

При объектно-ориентированном программировании (ООП) практически все компоненты компьютерной системы являются объектами. В Excel VBA *объектом* считается любой элемент приложения – ячейка, лист, рабочая книга, диаграмма. Фактически объектом является само приложение Microsoft Excel. Объекты могут включать области ячеек, рамки ячеек, окна, сценарии, стили и т. д. Каждый класс объектов имеет свое множество свойств, функций и событий.

Одинаковые объекты формируют *коллекцию*. Например, коллекция `WorkSheet` состоит из всех рабочих листов конкретной рабочей книги.

Когда в рабочий лист добавляется объект, то создается экземпляр класса объектов и объекту присваиваются те значения свойств, которые определены в этом классе. По умолчанию экземпляры одного и того же класса имеют имя, состоящее из имени родительского класса и порядкового номера.

Пример 5.1. На рабочую книгу с названием `Книга1.xls` можно сослаться следующим образом:

```
Application.Workbooks("Книга1.xls")
```

Это ссылка на рабочую книгу в коллекции Workbooks. Коллекция Workbooks находится в объекте Application. Переходя на следующий уровень, можно сослаться на Лист1 в книге Книга1.xls.

```
Application.Workbooks("Книга1.xls").Worksheets("Лист1")
```

Перейдя еще на один уровень ниже, можно сослаться на отдельную ячейку.

```
Application.Workbooks("Книга1.xls").Worksheets("Лист1").Range("A1")
```

5.1.2. Свойства

Свойство является атрибутом объекта, описывающим, как объект выглядит (цвет, размер и местоположение) и как он действует (является ли видимым, ссылается ли на другой объект). Когда создается объект, Microsoft Excel выполняет процедуру создания экземпляра этого объекта. Данная процедура, хранимая в классе объектов, присваивает значения всем свойствам, что позволяет сразу же работать с объектом. Для того чтобы в VBA присвоить объекту новое значение, необходимо лишь создать оператор присваивания, в котором слева от знака равенства будут указаны имя и свойство объекта (разделенные точкой), а справа – новое значение.

Пример 5.2. Указанный ниже оператор заменит присвоенное по умолчанию имя листа Лист1 именем Счета:

```
Лист1.Name=«Счета»
```

Свойство листа Name отображается на ярлычке листа. Причем данное свойство входит также во множество других классов.

5.1.3. Методы

Метод – это действие, которое может быть выполнено над объектом. Методы исполняются посредством выполнения процедуры метода, которая является членом класса объектов. Формат вызова метода: имя объекта и имя метода, разделенные точкой.

Пример 5.3. Чтобы использовать метод Protect для защиты объекта Лист1 от внесения изменений, следует использовать следующую команду:

```
Лист1.Protect
```

5.1.4. События

Всякий раз, когда пользователь взаимодействует с определенным объектом в рабочем листе, совершается событие. Каждый класс объектов имеет собственную группу событий, на которые объекты этого класса реагируют. Так, рабочая книга имеет событие *NewSheet*, которое выполняется тогда, когда пользователь добавляет в нее новый лист; событие *SelectionChange* происходит каждый раз при выборе пользователем другой ячейки или диапазона ячеек. Microsoft Excel снабжает каждую процедуру события первой и последней строкой; при желании можно написать программу для события, на которое должны реагировать объекты.

5.1.5. Взаимодействие объектов, свойств и методов

Когда создается лист в рабочей книге, функцией-членом в классе *WorkSheet* создается объект *WorkSheet*. Значения свойств нового объекта присваиваются по умолчанию и с объектом можно сразу же работать. К этому объекту можно применить любой из встроенных в класс методов, таких как *PrintOut* или *Save*, создать программу реагирования на такие события, как *SheetCalculate* или *Open*.

5.1.6. Аналогия

Для того чтобы пояснить все вышесказанное об объектах, свойствах, методах и событиях, приведем некоторую аналогию с объектами, которые нас окружают. В этой аналогии Excel сравнивается с сетью ресторанов быстрого питания.

Основной элемент Excel – объект *Workbook* (*Рабочая книга*). В сети ресторанов быстрого питания основным элементом является отдельный ресторан. В Excel можно добавлять рабочие книги и закрывать их; все открытые рабочие книги называются *Workbook* (коллекция объектов *Workbooks*). Аналогичным образом руководство сети ресторанов может открывать новые рестораны и закрывать старые – и все они в сети могут рассматриваться как коллекция объектов *Restaurants*.

Рабочая книга Excel является объектом, но она также содержит другие объекты, например, рабочие листы, диаграммы, модули VBA и т.д. Более того, каждый объект в рабочей книге может содержать собственные объекты. Например, объект *Worksheet* (*Рабочий лист*) включает объекты *Range* (*Диапазон*), *PivotTable* (*Сводная таблица*), *Shape* (*Форма*) и т.д.

Продолжим нашу аналогию: ресторан быстрого питания (как и рабочая книга) содержит свои объекты, например, кухню *Kitchen*, столовое помещение *DiningArea* и *Tables* (коллекция столов). Кроме того, руководство вправе добавлять или удалять объекты из объекта *Restaurant*. Например, в коллекцию *Tables* можно добавить дополнительные столы. Каждый такой объект иногда содержит другие объекты. Например, объект *Kitchen* включает объект *Stove* (*Плита*), *VentilationFan* (*Вентилятор*), *Chef* (*Шеф-повар*), *Sink* (*Раковина*) и т.д.

Объекты Excel обладают свойствами. Например, объект *Range* имеет свойства значения *Value* и имени *Name*, а объект *Shape* – свойства ширины *Width*, высоты *Height* и т.д. Объекты в ресторане быстрого питания тоже обладают свойствами. К примеру, объект *Stove* имеет такие свойства, как температуру *Temperature* и количество конфорок *NumberOfBurners*. У объекта *VentilationFan* есть собственный набор свойств (*TurnedOn* (*Включен*), *RPM* (*Частота вращения*) и т.д.).

Помимо свойств, объекты Excel также располагают методами, выполняющими операции над объектом. Например, метод *ClearContents* удаляет содержимое объекта *Range*. Объекты ресторана быстрого питания тоже обладают методами. Можно легко представить себе метод *ChangeThermostat* (*Изменить температуру*) объекта *Stove* или метод *SwitchOn* (*Включить*) для объекта *VentilationFan*.

В Excel методы иногда используются для изменения свойств объекта. Метод *ClearContents* объекта *Range* изменяет свойство *Value* объекта *Range*. Аналогично, метод *ChangeThermostat* объекта *Stove* изменяет его свойство *Temperature*.

В VBA существует возможность писать программы для управления объектами Excel. В ресторане быстрого питания руководство может давать указания по работе с объектами в ресторанах («Включить плиту и переключить вентилятор на максимальный режим»).

5.2. Компоненты редактора Visual Basic

Начиная с Excel 97, VBA-модули вводятся с помощью специальной программы, включенной в приложение Excel. Для работы и просмотра модулей VBA используется редактор Visual Basic (*VBE – Visual Basic Editor*). Он запускается как автоматически – при редактировании макросов, так и отдельно – для создания новых процедур. Редактор VBE обладает всеми необходимыми средствами для управления VBA-кодом в Excel.

5.2.1. Запуск VBE

Во время работы в Excel перейти к окну VBE можно одним из следующих способов:

– нажать Alt+F11;

– выполнить команду *меню Сервис→Макрос→Редактор Visual Basic*;

– щелкнуть на кнопке «Редактор Visual Basic»  , расположенной на панели инструментов Visual Basic (рис. I.5.1.).

Рис. I.5.1. Панель инструментов Visual Basic

5.2.2. Окна VBE

VBE состоит из следующих элементов (рис. I.5.2):

– *строка меню*. Строка меню VBE работает, как и строка меню любого другого приложения. Она содержит команды, используемые для управления различными компонентами VBE. Кроме того, для выполнения многих команд меню используются сочетания клавиш. Например, для команды *меню View → Immediate Window* (*меню Вид → Окно отладки*) применяется комбинация клавиш Ctrl+G.

В VBE также представлены контекстные меню. Щелкнув правой кнопкой мыши практически на любом элементе окна VBE, отобразится меню, предлагающее ряд команд;

– *панели инструментов.* Стандартная панель инструментов «Standard», которая по умолчанию находится под строкой меню, – это одна из панелей инструментов, используемых в VBE. Панели инструментов VBE работают, как и в Excel: можно задавать специальные настройки для панелей инструментов, перемещать их, отображать другие панели инструментов и т.д. Для управления панелями инструментов VBE используется команда *меню View → Toolbars → Customize* (*меню Вид → Панели инструментов → → Настройка*);

– окно «Project Explorer». В окне «Project Explorer» отображается древовидная структура всех открытых в данный момент в Excel рабочих книг (включая надстройки и скрытые рабочие книги). Каждая рабочая книга известна как проект.

Если в редакторе Visual Basic окно «Project Explorer» не отображено, то его можно вывести на экран с помощью сочетания клавиш Ctrl+R. Чтобы скрыть его, следует щелкнуть на кнопке закрытия окна в строке заголовка (или щелкнуть правой кнопкой мыши в любом месте окна и выбрать команду *Hide* (*Скрыть*) из контекстного меню);

– *окно кода*. Окно кода (которое иногда называют «Module») содержит код VBA. Для каждого элемента проекта представлено собственное окно кода. Чтобы просмотреть код объекта, нужно выполнить двойной щелчок мышью на этом объекте в окне «Project Explorer». Например, чтобы просмотреть код объекта `Лист1`, нужно выполнить двойной щелчок на элементе `Лист1` в окне «Project Explorer». Если для него VBA-код не создавался, открывшееся окно будет пустым.

Существует еще один способ просмотреть код объекта: нужно выделить этот объект в окне «Project Explorer», а затем щелкнуть на кнопке «View Code» («Просмотр кода») на панели инструментов вверху окна «Project Explorer».

Окна кода рассмотрены далее в разделе «Работа с окнами кода»;

– *окно отладки*. Окно отладки предназначено для непосредственного выполнения операторов VBA, тестирования операторов и отладки кода. Это окно может отображаться и скрываться. Если окно «Immediate» в данный момент не отображается на экране, то для его вывода на экран нужно нажать Ctrl+G. Чтобы закрыть окно «Immediate», следует щелкнуть на кнопке его закрытия в строке заголовка (или щелкнуть правой кнопкой мыши в любом месте окна и выбрать опцию *Hide* из контекстного меню).

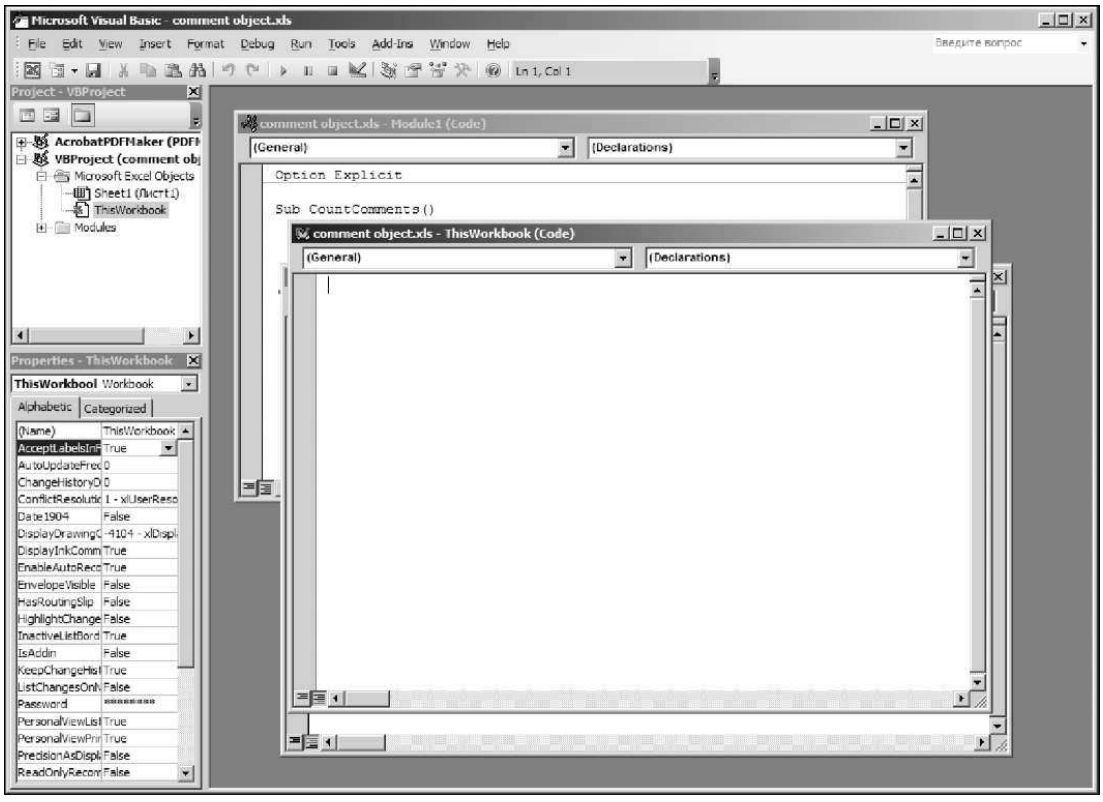


Рис. I.5.2. Окно Microsoft Visual Basic

5.2.3. Работа с «Project Explorer»

При работе в редакторе Visual Basic каждая рабочая книга Excel и открытые в данный момент надстройки рассматриваются как проекты. Проект можно считать коллекцией объектов, организованных в виде иерархической структуры. Проект можно развернуть, если щелкнуть на знаке «плюс» слева от его названия в окне «Project Explorer». Проект сворачивается при щелчке на знаке «минус» слева от его названия. Кроме того, для разворачивания и сворачивания проекта можно использовать кнопку «Toggle Folders» («Показать папки») на панели инструментов окна «Project Explorer». При попытке развернуть проект, защищенный паролем, отображается окно ввода пароля.

На рисунке I.5.3. показано окно «Project Explorer» с несколькими проектами (среди них только одна рабочая книга).

Рис. I.5.3. Окно Project Explorer с несколькими проектами

Дерево каждого проекта в развернутом виде имеет как минимум один узел под названием «Microsoft Excel Objects». В этом узле содержатся элементы каждого рабочего листа и лист диаграмм рабочей книги (рабочий лист считается объектом), а также объект под названием «ЭтаКнига», представляющий объект «ActiveWorkbook».

Если в проекте используются модули VBA, то в дереве отображается также узел «Modules», в котором перечислены модули. Проект может также содержать узел «Forms», содержащий объекты «UserForm» (пользовательские формы, известные как пользовательские диалоговые окна). Если в проекте находятся модули классов, то в дереве отображается узел под названием «Class Modules».

5.2.4. Добавление нового модуля VBA

Чтобы добавить в проект новый модуль VBA, нужно выделить название проекта в окне «Project Explorer» и выполнить команду *меню Insert → Module* (*меню Вставка → Модуль*). Также можно щелкнуть правой кнопкой мыши на названии проекта и выбрать команду *Insert → Module* в контекстном меню.

При записи макроса Excel автоматически вставляет модуль VBA для хранения записанного кода.

Удаление модуля VBA. Чтобы удалить из проекта модуль VBA или модуль класса, нужно выделить название модуля в окне «Project Explorer» и воспользоваться командой *меню File → Remove xxx*, где *xxx* – название модуля. Кроме того, можно щелкнуть правой кнопкой мыши на названии модуля и выбрать команду *Remove xxx* из контекстного меню. Невозможно удалить программные модули, соответствующие рабочей книге (программный модуль «ЭтаКнига»), а также рабочему листу (например, программный модуль `Лист1`).

5.2.5. Работа с окнами кода

Каждому объекту в проекте соответствует свое окно кода. Такими объектами могут быть:

– сама рабочая книга («ЭтаКнига» в окне «Project Explorer»);

– рабочий лист или лист диаграмм рабочей книги (например, `Лист1` или «Диаграмма1» в окне «Project»);

– модуль VBA;

– модуль класса (специальный тип модуля, позволяющий создавать новые классы объектов);

– форма «UserForm».

5.2.6. Сохранение программы VBA

Как правило, окно кода содержит четыре типа кода:

– процедуры. Это набор инструкций, выполняющих определенное действие;

– функции. Это набор инструкций, возвращающий значение или массив значений (концепция функции VBA подобна такой же функции Excel);

– процедуры свойств. Специальные процедуры, используемые в модулях классов;

– объявление – это информация о переменной, предоставляемая VBA. Например, можно объявить тип данных для переменных, которые планируется использовать в коде.

В отдельном модуле VBA может храниться любое количество процедур, функций и объявлений. Способ организации модуля VBA зависит от желания пользователя:

– можно записать весь код VBA приложения в одном модуле VBA;

– можно разделить код на несколько разных модулей.

5.2.7. Терминология

В настоящем УМК применяются термины *подпрограмма*, *процедура*, *макрос*. Программисты для описания автоматизированной задачи обычно используют слово *процедура*. В Excel процедуру также называют *макросом*. Технически процедура может быть двух видов: процедура типа *Sub* или процедура *функции* (или просто *функция*), оба вида иногда называют *подпрограммами*. В УМК эти термины будут использоваться как синонимы.

Несмотря на то, что пользователям предоставляются широкие возможности по определению места хранения кода VBA, существуют некоторые ограничения на его расположение. Процедуры обработки событий должны содержаться в окне кода объекта, которому соответствует это событие. Например, если пишется процедура, которая выполняется при открытии рабочей книги, то эта процедура должна располагаться в окне кода для объекта «ЭтаКнига» и иметь специальное название.

5.2.8. Способы ввода кода VBA

Для того чтобы выполнить одно из действий программным образом, необходимо написать программу VBA в окне кода. Код VBA располагается в процедуре. Процедура состоит из операторов VBA. На данном этапе (для примера) остановимся только на одном типе окна кода – модуле VBA.

Добавить код в модуль VBA можно тремя способами:

– ввести код традиционным способом: с помощью клавиатуры;

– использовать функцию записи макросов в Excel, чтобы записать действия и преобразовать их в код VBA (запись макросов подробно рассматривалась в части I УМК «Компьютерные информационные технологии»);

– скопировать текст программы из другого модуля и вставить его в модуль, над которым ведется работа.

Ввод кода вручную. Непосредственное введение кода выполняется с помощью клавиатуры. Клавиша Tab при этом поможет задать отступ в строках, которые логически принадлежат одной группе (например, условные операторы `If` и `End If`).

Ввод и редактирование кода в модуле VBA выполняется обычным образом. Можно выделять текст, копировать, вырезать его, а затем вставлять в другое место программы.

Отдельная инструкция в VBA может иметь произвольную длину. Однако для обеспечения удобочитаемости кода длинные инструкции лучше разбить на две или более строки. Для этого следует в конце строки ввести пробел и символ подчеркивания, а затем нажать Enter и продолжить инструкцию в следующей строке.

Как и в Excel, в VBE существует несколько уровней отмены выполненных операций. Поэтому, если вы по ошибке удалили инструкцию, можете несколько раз щелкнуть на кнопке «Undo» («Отменить») или нажать Ctrl+Z, и инструкция вновь появится в коде. После отмены операции можно щелкнуть на кнопке «Redo» («Вернуть»), чтобы вернуть изменения, которые ранее были отменены.

При вводе кода VBE вносит некоторые изменения во введенный текст. Например, если был пропущен пробел перед или после знака равенства (=), VBE вставит его автоматически. Кроме того, изменяется цвет некоторых слов кода.

Пример 5.4. Создание и выполнение простейшей процедуры `SayHello` на VBA.

Для создания простейшей процедуры нужно добавить в проект модуль VBA, а затем ввести следующую процедуру в окне кода данного модуля (рис. I.5.4):

```
Sub SayHello()
   Msg = "Ваше имя " & Application.UserName & "?"
   Ans = MsgBox(Msg, vbYesNo)
   If Ans = vbNo Then
   MsgBox "Ничего страшного."
   Else
   MsgBox "Наверное, я ясновидящий!"
   End If
End Sub
```

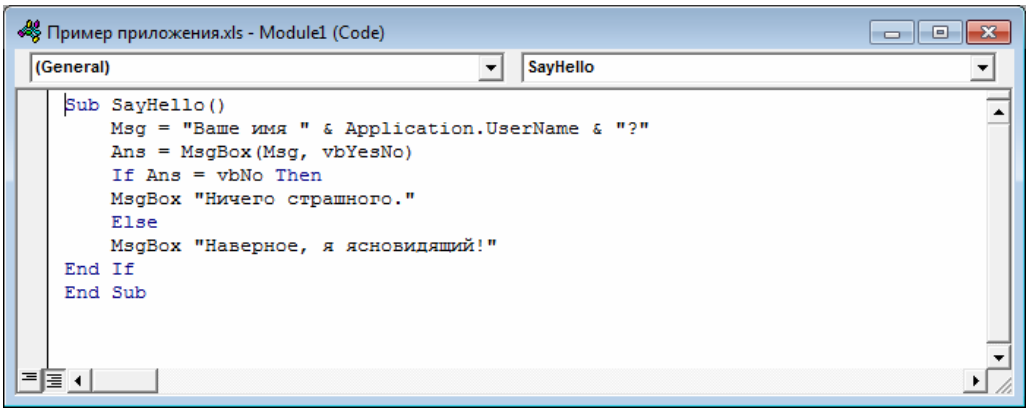


Рис. I.5.4. Простейшая процедура на VBA

Для выполнения процедуры `SayHello` необходимо убедиться, что курсор находится во введенном тексте. Затем выполнить одно из следующих действий:

– нажать функциональную клавишу F5;

– выполнить команду *меню Run → Run Sub/UserForm*;

– щелкнуть на кнопке «Run Sub/UserForm» на стандартной панели инструментов.

Если код введен правильно, процедура будет выполнена, и появится простое диалоговое окно, в котором отображается имя пользователя, заданное в диалоговом окне «Параметры» (рис. I.5.5, I.5.6). Здесь можно выбрать ответ. Обратите внимание, что при выполнении макроса активизируется программа Excel.

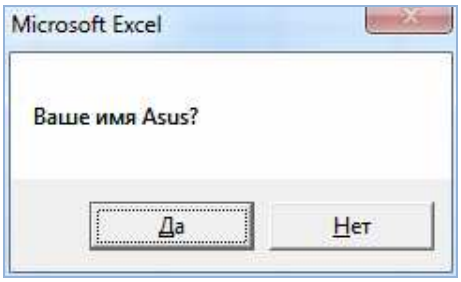
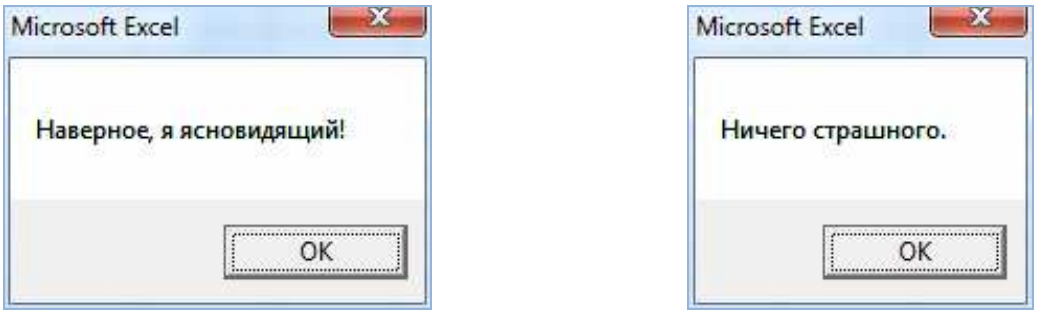


Рис. I.5.5. Окно диалога для выбора ответа



а б

Рис. I.5.6. Возможные результаты выполнения созданной процедуры

Написанная только что программа представляет собой процедуру VBA (называемую также макросом). При создании этой простой процедуры использовались следующие принципы (все они рассмотрены далее в разделе 6):

- объявление процедуры (первая строка);
- присвоение значения переменным (`Msg` и `Ans`);
- конкатенация строк (с помощью оператора `&`);
- использование встроенной функции VBA (`MsgBox`);
- применение встроенных констант VBA (`vbYesNo` и `vbNo`);
- использование конструкции `If-Then-Else`;
- окончание процедуры (последняя строка).

5.3. Настройка среды VBE

Чтобы сделать редактор VBE более удобным, рекомендуется настроить некоторые его параметры.

Для этого нужно выполнить команду *меню Tools → Options* (*меню Сервис → Параметры*). Появится диалоговое окно «Options» («Параметры») с четырьмя вкладками: «Editor» («Редактор»), «Editor Format» («Формат редактора»), «General» («Общие») и «Docking» («Присоединение»).

5.3.1. Вкладка «Editor»

На рисунке I.5.7 показаны параметры, доступные на вкладке «Editor» диалогового окна «Options».

Параметр «Auto Syntax Check». Настройка «Auto Syntax Check» («Автоматическая проверка синтаксиса») определяет, будет ли появляться диалоговое окно, когда VBE обнаруживает синтаксическую ошибку в коде VBA. В этом диалоговом окне указывается тип допущенной ошибки. Если снять этот флажок, то VBE выделит синтаксические ошибки, отобразив соответствующие фрагменты кода другим цветом, и не будет необходимости работать в диалоговых окнах, которые появляются на экране.

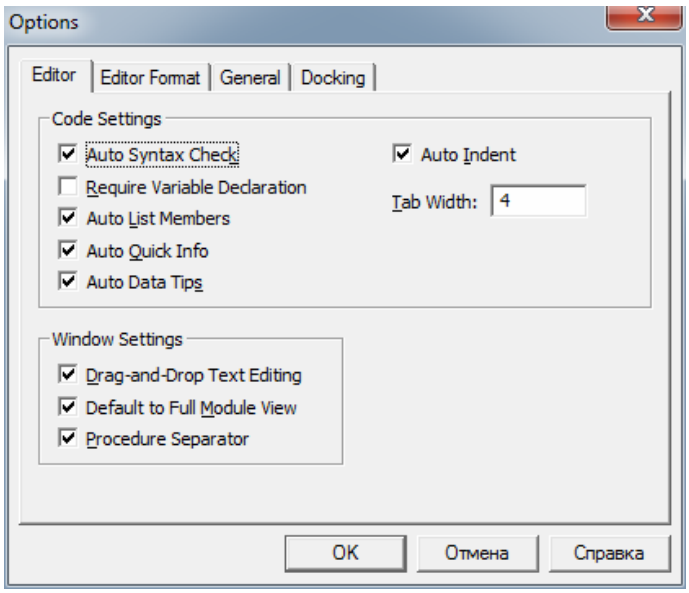


Рис. I.5.7. Вкладка «Editor» диалогового окна «Options»

Параметр «Require Variable Declaration». При установленном параметре «Require Variable Declaration» («Обязательное декларирование переменных») VBE вставляет в начале каждого нового модуля оператор Option Explicit. Если в модуле задан этот оператор, то нужно явно определить каждую используемую в нем переменную. Если переменные не объявляются, все они имеют тип данных *Variant*; это достаточно гибко, но неэффективно с точки зрения использования аппаратных ресурсов и скорости выполнения кода.

Изменение параметра «Require Variable Declaration» затрагивает только новые модули, а не уже существующие.

Параметр «Auto List Members». Если выставлена опция «Auto List Members» («Автоматическая вставка объектов»), VBE предоставляет помощь при вводе кода VBA, отображая список элементов текущего объекта. К этим элементам относятся методы и свойства объекта, название которого вводится вручную.

Данный параметр весьма полезен, поэтому его рекомендуется всегда активизировать. На рисунке I.5.8 показан пример использования опции «Auto List Members». В данном примере VBE отображает список элементов объекта «Application». Можно выбрать элемент из списка, чтобы не вводить его с помощью клавиатуры (в результате название элемента будет введено без ошибок).

Если включен параметр «Auto Quick Info» («Отображать краткие сведения»), VBE будет отображать информацию об аргументах функций, свойств и методов, названия которых вводятся с клавиатуры. Это очень полезно, поэтому рекомендуется всегда оставлять эту настройку включенной. На рисунке I.5.9 данная функция показана в действии: отображается синтаксис свойства «Range».

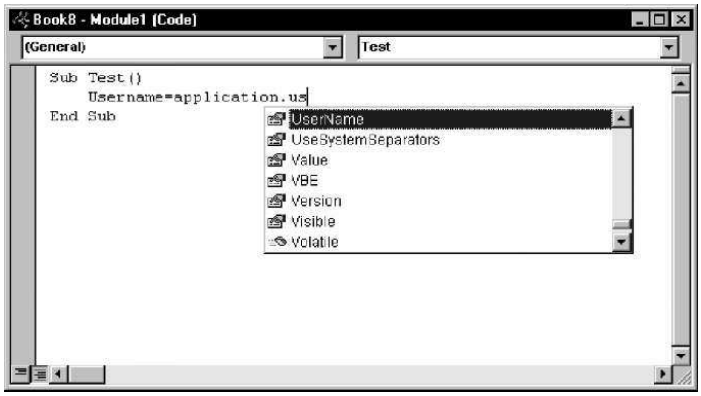


Рис. I.5.8. Пример использования опции Auto List Members

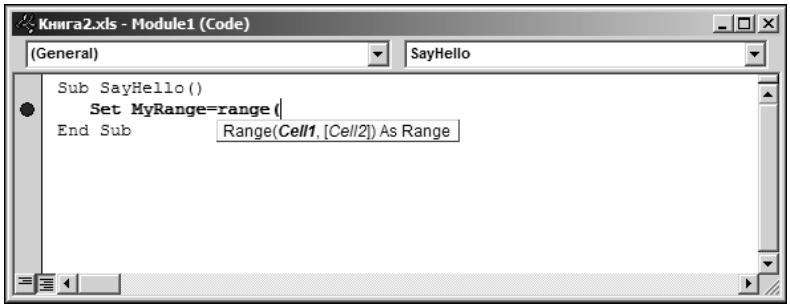


Рис. I.5.9. Пример функции Auto Quick Info

Параметр «Auto Data Tips». Если включен параметр «Auto Data Tips», VBE отображает при отладке кода значение переменной, над которой находится указатель мыши.

Параметр «Auto Indent». Настройка «Auto Indent» («Автоматический отступ») определяет, располагает ли автоматически VBE каждую новую строку программы с тем же отступом, который задан для предыщущей строки. Тем, кто использует отступы в программных кодах, рекомендуется всегда обращаться к этому параметру. Можно также задать количество символов в отступе (по умолчанию указано значение 4).

Параметр «Drag-and-Drop Text Editing». При выборе параметра «Drag-and-Drop Text Editing» («Включить редактирование перетаскиванием») можно копировать и перемещать текст, перетаскивая его с помощью мыши.

Параметр «Default to Full Module View». Параметр «Default to Full Module View» («По умолчанию использовать полный режим просмотра») определяет принцип просмотра процедуры. Если он включен, процедуры в окне кода помещаются в одно окно с полосой прокрутки. Если же он отключен, то можно просмотреть в определенный момент только одну процедуру.

Параметр «Procedure Separator». Когда параметр «Procedure Separator» («Разделение процедур») включен, в конце каждой процедуры в окне кода отображаются специальные разделители.

5.3.2. Вкладка «Editor Format»

На рисунке I.5.10 показана вкладка «Editor Format» диалогового окна «Options».

Параметр «Code Colors». Параметр «Code Colors» («Цвета кода») предоставляет возможность выбрать цвета кода (текста и фона) и индикатора, который используется для выделения разных элементов программы VBA. Цвета выбираются в зависимости от личных предпочтений.

Параметр «Font». Параметр «Font» («Шрифт») предоставляет возможность указать шрифт, используемый в модулях VBA. Наибольшая эффективность достигается при работе с моноширинным шрифтом (например, Courier New). В таком шрифте все символы имеют одинаковую ширину, что делает программу более удобной для восприятия и анализа, так как все символы одинаково выровнены, кроме того, хорошо видны пробелы между словами.

Список «Size». Список «Size» («Размер») определяет размер шрифта кода модулей VBA. Эта настройка зависит от личных предпочтений, которые, в свою очередь, определяются разрешением монитора и особенностями зрения. По умолчанию размер задан равным 10 пт.

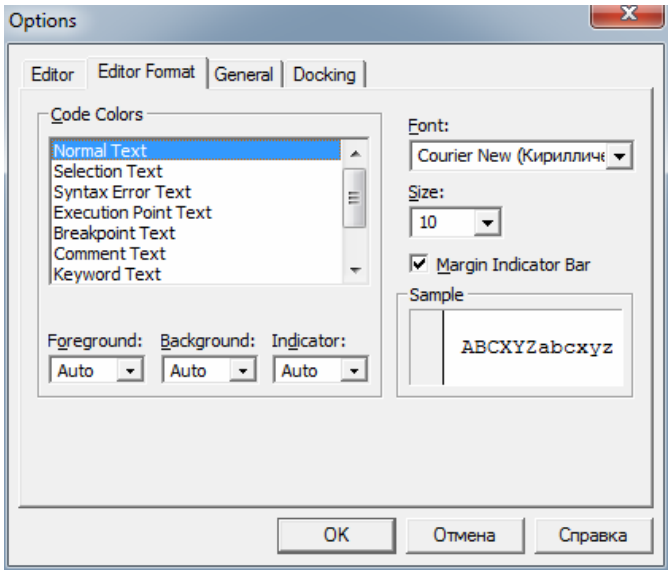


Рис. I.5.10. Вкладка «Editor Format» диалогового окна «Options»

Параметр «Margin Indicator Bar». Этот параметр отображает вертикальную полосу вдоль левой границы окна кода, на которой высвечиваются всевозможные индикаторы. Его необходимо выставить, в противном случае не будут видны полезные графические извещения при отладке кода.

5.3.3. Вкладка «General»

На рисунке I.5.11 показаны параметры, доступные на вкладке «General» («Общие») диалогового окна «Options». Практически во всех случаях идеально использовать настройки по умолчанию.

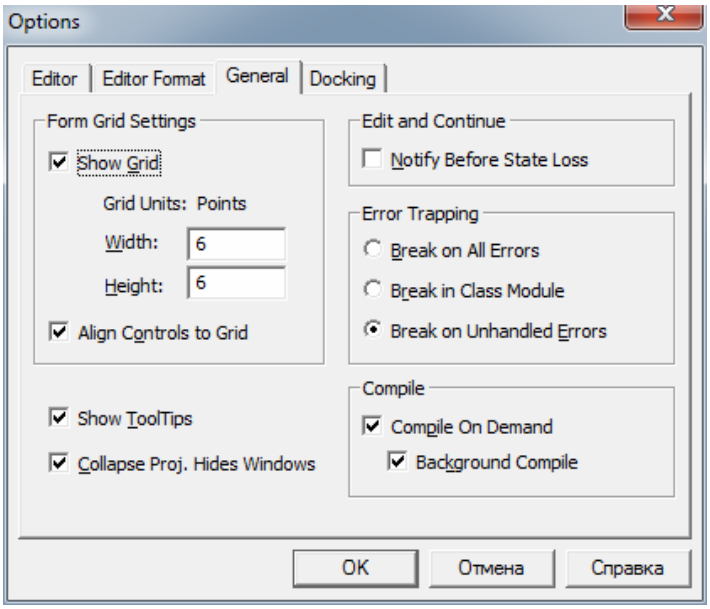


Рис. I.5.11. Вкладка «General» диалогового окна «Options»

Раздел «Error Trapping» («Захват ошибок») определяет, что происходит при возникновении ошибки. Если создается процедура обработки ошибок, то следует убедиться, что включен переключатель «Break on Unhandled Errors» («Остановка при возникновении неисправимой ошибки»). При заданном параметре «Break on All Errors» («Остановка при возникновении любой ошибки») процедуры обработки ошибок игнорируются.

5.3.4. Вкладка «Docking»

На рисунке I.5.12 показана вкладка «Docking» диалогового окна «Options». Ее параметры определяют поведение нескольких окон редактора VBE – отображаются окна, которые могут быть прикреплены. Когда окно прикреплено, оно фиксируется по отношению к одной из границ окна VBE. В результате намного легче найти вспомогательное окно, так как оно отображается в строго определенной области. Если отключить все параметры прикрепления, то окна перемешаются между собой, а это усложнит работу. Как правило, идеальным выбором будут настройки по умолчанию.

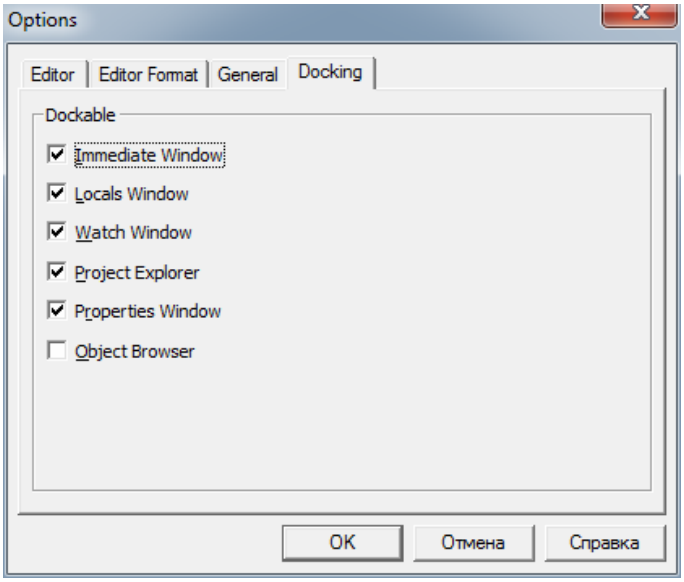


Рис. I.5.12. Вкладка «Docking» диалогового окна «Options»

6. Программирование на VBA

6.1. Элементы языка VBA. Обзор

В этом разделе будут проанализированы *элементы языка VBA*, ключевые слова и операторы, используемые для написания процедур VBA.

Для начала в качестве примера рассмотрим простую процедуру. Она хранится в модуле VBA и вычисляет сумму первых 100 целых чисел. По окончании вычислений процедура отображает сообщение с результатом.

```
Sub VBA_Demo()
   'Это пример простой процедуры VBA
   Dim Total As Integer, i As Integer
   Total = 0
   For i = 1 To 100
          Total = Total + i Next i
   MsgBox Total End
Sub
```

В данной процедуре используются некоторые популярные элементы языка, в том числе комментарий (строка после апострофа), переменная (`Total`), два оператора присвоения (`Total = 0` и `Total = Total + i`), циклическая структура (`For-Next`) и оператор VBA (`MsgBox`). Все указанные элементы рассматриваются далее в этом разделе.

Процедуры VBA не всегда управляют объектами. Например, рассмотренная выше процедура не имеет ничего общего с объектами – она оперирует цифрами.

6.1.1. Комментарии

Комментарий – это описательный текст, который включается в код. VBA полностью игнорирует текст комментария. Комментарии можно использовать самым произвольным образом для описаний действий в коде (предназначение оператора не всегда очевидно).

Для комментария можно использовать новую строку либо вставить комментарий после инструкции в той же строке. Комментарий обозначается апострофом. Кроме того, для обозначения строки комментария можно использовать ключевое слово `Rem`. В отличие от апострофа, `Rem` используется только в начале строки, его не допускается применять в той же строке, что и инструкция.

6.1.2. Рекомендации по вводу кода VBA

Код VBA, который содержится в модуле VBA, состоит из инструкций. Общепринято вводить по одной инструкции в каждой строке. Однако этот стандарт не является обязательным требованием, и для разделения нескольких инструкций в одной строке можно применить двоеточие. В следующем примере четыре инструкции приведены в одной строке.

```
Sub OneLine()
   x=1: y=2: z=3: MsgBox x + y + z
End Sub
```

Строка кода может иметь любую длину; модуль VBA продолжается на следующей строке, когда текущая строка доходит до правой границы окна. В длинных строках допускается использование оператора продолжения строки VBA: пробел с подчеркиванием (_).

```
Sub LongLine()
   SummedValue = _
               Worksheets("Лист1").Range("A1").Value + _
               Worksheets("Лист2").Range("A1").Value
End Sub
```

При записи макросов Excel довольно часто символы подчеркивания применяют для разбиения длинных операторов на несколько строк.

После ввода инструкции редактор Visual Basic выполняет следующие действия в целях улучшения читабельности кода:

– вставляет пробелы между операторами. К примеру, если ввести `Ans=1+2` (без пробелов), то VBE преобразует это выражение следующим образом:

```
Ans = 1 + 2
```

– VBA изменяет регистр символов ключевых слов, свойств и методов. Если ввести выражение `Result=activesheet.range("a1").value=12`, то VBA преобразует его в следующий оператор:

```
Result = ActiveSheet.Range("a1").Value = 12
```

Обратите внимание, что текст внутри кавычек (в данном случае, `"a1"`) не изменяется;

– так как названия переменных VBA не чувствительны к регистру, то интерпретатор по умолчанию изменяет названия всех переменных, состоящих из букв одного регистра, таким образом, чтобы их регистр соответствовал последнему введенному варианту. Например, если сначала определить переменную как `myvalue` (все буквы в нижнем регистре) и затем ввести переменную `MyValue` (смешанный регистр), то VBA поменяет название переменной во всех остальных случаях на `MyValue`. Исключение может быть лишь тогда, когда переменная объявляется с помощью ключевого слова `Dim` или другого специального оператора – название переменной останется неизменным, в виде, объявленном в начале процедуры.

– VBE просматривает инструкции на наличие синтаксических ошибок. Если VBE находит ошибку, то цвет строки изменяется, а на экране может быть отображено сообщение, описывающее проблему.

Панель инструментов «Edit» (рис. I.6.1) в VBE содержит несколько полезных кнопок:

– выделив группу инструкций, а затем, нажав кнопку «Comment Block» , можно преобразовать инструкции в комментарии;

– кнопка «Uncomment Block» преобразовывает группу комментариев обратно в инструкции.

Рис. I.6.1. Панель инструментов «Edit»

6.1.3. Переменные, типы данных и константы

Главное предназначение VBA – обработка данных. Некоторые данные сохраняются в объектах, например, диапазонах рабочих листов. Другие данные хранятся в созданных разработчиком *переменных*.

Переменная представляет собой именованное место хранения данных в памяти компьютера. Переменные могут содержать данные разных *типов* – от простых логических, или булевых, значений (`True` или `False`) до больших значений с двойной точностью (см. следующий раздел). Значение присваивается переменной с помощью оператора равенства (подробнее об этом – далее в разделе).

Существует ряд правил именования переменных в VBA:

– можно использовать в названиях символы букв, числа и некоторые знаки препинания, но первой в имени переменной всегда должна быть буква;

– VBA не различает регистры. Чтобы сделать имена переменных удобочитаемыми, программисты часто используют смешанный регистр (например, *InterestRate*, а не *interestrate*);

– нельзя использовать в именах пробелы или точки. Чтобы сделать имена переменных более удобными для чтения, программисты вводят символ подчеркивания (*Interest_Rate*);

– специальные символы объявления типов (#, $, %, & или !) также не применяются в имени переменной;

– названия переменных ограничены длиной в 254 символа.

В приведенном далее списке содержатся отдельные примеры выражений присвоения, в которых используются различные типы переменных. Названия переменных указываются слева от знака равенства. Каждый оператор присваивает переменной слева от знака равенства значение, которое располагается справа от знака равенства.

```
x =1
InterestRate = 0.075
LoanPayoffAmount = 234089
DataEntered = False
x = x + 1
MyNum = YourNum * 1.25
UserName = "Bob Johnson"
DateStarted = #3/14/98#
```

В VBA используется очень много зарезервированных слов, т.е. таких слов, которые не допускается применять в качестве названий переменных или процедур. Если попытаться ввести одно из таких слов, то будет отображено сообщение об ошибке. Например, несмотря на то, что зарезервированное слово `Next` могло бы стать описательным названием многих переменных, следующая инструкция генерирует синтаксическую ошибку.

```
Next = 132
```

Определение типов данных. VBA автоматически обрабатывает любые типы данных. *Тип данных* указывает, в каком виде данные хранятся в памяти: как целые значения, действительные числа, текст и т.п.

В таблице I.6.1. перечислены поддерживаемые в VBA типы данных.

Таблица I.6.1

Встроенные типы данных VBA

Тип данных	Резервируется байт	Диапазон значений
`Byte`	1 байт	От 0 до 255
`Boolean`	2 байта	`True` (Истина) или `False` (Ложь)
`Integer`	2 байта	От -32768 до 32767
`Long`	4 байта	От -2147483648 до 2147483647
`Single`	4 байта	От -3,402823E38 до -1,401298E-45 (для отрицательных значений); от 1,401298E-45 до 3,402823E38 (для положительных значений)
`Double`	8 байт	От -1,79769313486232E308 до -4,94065645841247E-324 (отрицательные числа); от 4,94065645841247E-324 до 1,79769313486232E308 (положительные числа)
`Currency`	8 байт	От -922337203685477,5808 до 922337203685477,5807

Окончание таблицы I.6.1.

Тип данных	Резервируется байт	Диапазон значений
`Decimal`	14 байт	+/-79228162514264337593543950335 без десятичных знаков; +/-7,9228162514264337593543950335 с 28-ю знаками после запятой
`Date`	8 байт	С 1 января 100 года до 31 декабря 9999 года
`Object`	4 байта	Любая ссылка на объект
`String` (переменной длины)	10 байт + длина строки	От 0 до приблизительно 2 млрд.
`String` (фиксированной длины)	Длина строки	От 1 до приблизительно 65400
`Variant` (числа)	16 байт	Любое числовое значение в рамках диапазона типа данных `Double`
`Variant` (символы)	22 байта + длина строки	От 0 до приблизительно 2 млрд.
Пользовательский	Зависит от типа	Зависит от элемента

Рекомендуется выбирать тот тип данных, в котором используется минимальное количество байт для хранения значений, но он также должен быть достаточным для представления максимальных значений переменных. При работе с данными в VBA скорость выполнения операций зависит от объема данных, которые обрабатываются с помощью VBA. Другими словами, чем меньше байт зарезервировано под данные, тем быстрее VBA получает доступ к данным и обрабатывает их.

Для проведения математических вычислений в рабочих листах Excel использует тип данных `Double`. Его советуем применять и в процессе обработки чисел в VBA для обеспечения той же точности вычислений. Для обработки целочисленных значений идеально подходит тип `Integer`, если, конечно, вы уверены, что используемые значения не превышают 32767. В противном случае обратитесь к типу данных `Long`. При управлении номерами строк в рабочем листе Excel лучше применять тип данных `Long`, так как количество строк в рабочем листе превышает максимальное значение, допустимое в типе данный `Integer`.

6.1.4. Объявление переменных

Если тип данных для переменной, используемой в процедуре VBA, не объявлен, по умолчанию будет задан тип данных `Variant`. Данные с типом `Variant` изменяют свой тип в зависимости от того, какие операции над ними выполняются. В следующей процедуре показано, каким образом переменная принимает разные типы данных.

```
Sub VariantDemo()
   MyVar = "12 3"
   MyVar = MyVar / 2
   MyVar = "Ответ: " & MyVar
   MsgBox MyVar
End Sub
```

В процедуре `VariantDemo` переменная `MyVar` вначале выступает строкой из трех символов. Затем эта «строка» делится на два и приобретает числовой тип данных. После этого `MyVar` присоединяется к строке, что вызывает обратное преобразование `MyVar` в строку. Оператор `MsgBox` отображает окончательное строковое значение: *Ответ: 61,5*.

Чтобы проиллюстрировать проблемы, которые могут возникнуть при обработке типа данных `Variant`, рассмотрим следующую процедуру:

```
Sub VariantDemo2()
   MyVar = "123"
   MyVar = MyVar + MyVar
   MyVar = "Ответ: " & MyVar
   MsgBox MyVar
End Sub
```

При выполнении этой процедуры в окне сообщений появится сообщение *Ответ: 123123*. Этот результат предопределен тем, что в процессе управления текстовыми данными, представленными типом `Variant`, оператор «+» выполняет конкатенацию (объединение) строк.

Определение типа данных. Для определения типа данных переменной используется функция VBA `TypeName`. Ниже представлена модифицированная версия предыдущей процедуры. Эта процедура на каждом шаге отображает тип данных переменной `MyVar`. Здесь видно, что сначала переменная является строкой, затем преобразовывается в числовой тип `Double` и в завершение снова становится строкой.

```
Sub VariantDemo2()
   MyVar = "123"
   MsgBox TypeName(MyVar)
   MyVar = MyVar / 2
   MsgBox TypeName(MyVar)
   MyVar = "Ответ: " & MyVar
   MsgBox TypeName(MyVar)
   MsgBox MyVar
End Sub
```

Благодаря VBA преобразование типов для необъявленных переменных выполняется автоматически. Однако при этом уменьшается скорость обработки данных и быстрее заполняется свободная память.

Таким оюразом, объявление переменных предоставляет два основных преимущества:

– программы работают быстрее и используют память более эффективно. Тип данных (по умолчанию `Variant`), резервирует больше памяти чем это необходимо и вызывает многократную проверку данных, занимающую процессорное время. Если программа точно знает тип данных, она не выполняет дополнительную проверку данных и резервирует ровно столько памяти, сколько необходимо для хранения конечных данных;

– объявление переменных позволяет избежать ошибок, связанных с неправильным вводом имен переменных.

Обязательное объявление всех переменных. Чтобы обеспечить обязательное объявление всех используемых переменных, необходимо включить следующую строку в качестве первой инструкции в модуле VBA.

```
Option Explicit
```

Этот оператор отвечает за то, что программа будет приостанавливаться всякий раз при нахождении в ней необъявленной переменной. VBA выведет сообщение об ошибке; для продолжения выполнения процедуры необходимо будет объявить переменную.

Область действия переменных. *Область действия* переменной определяет, в каких модулях и процедурах она может использоваться. Существующие области действия переменных описаны в таблице I.6.2.

Таблица I.6.2

Области действия переменных

Область действия	Способ объявления переменной
Отдельная процедура	В процедуру включается оператор `Dim` или `Static`
Отдельный модуль	Перед первой процедурой в модуле вводится оператор `Dim` или `Private`
Все модули	Перед первой процедурой в модуле вводится оператор `Public`

Локальные переменные. *Локальная переменная* – это переменная, объявленная в процедуре. Локальные переменные могут использоваться только в процедуре, в которой они объявлены. После выполнения процедуры переменная становится невостребованной, поэтому Excel освобождает соответствующую область памяти.

Наиболее часто используемым способом объявить локальную переменную является вставка оператора `Dim` между операторами `Sub` и `End Sub`. Операторы `Dim` обычно вводятся непосредственно после оператора `Sub`, перед кодом процедуры.

В представленной далее процедуре используется шесть локальных переменных, объявленных с помощью операторов `Dim`. Последний оператор `Dim` в этом примере объявляет не тип данных, а только саму переменную. В результате переменная приобретает тип `Variant`.

```
Sub MySub()
   Dim x As Integer
   Dim First As Long
   Dim InterestRate As Single
   Dim TodaysDate As Single
   Dim UserName As String * 2 0
   Dim MyValue
   ' -[Здесь указывается код процедуры] -
End Sub
```

Кроме того, можно объявить несколько переменных, воспользовавшись одним оператором `Dim`.

```
Dim x As Integer, y As Integer, z As Integer
Dim First As Long, Last As Double
```

Другой способ указания типов данных для переменных.

Язык VBA позволяет присоединить символ к названию переменной, чтобы указать ее тип данных. Например, можно объявить переменную `MyVar` как целое число, добавив к ее названию символ %.

```
Dim MyVar%
```

Символы объявления типов данных представлены для большинства типов данных VBA (отсутствующие в таблице I.6.3 типы данных не имеют собственного символа объявления типа).

Таблица I.6.3

Символы объявления типов данных

Тип данных	Символ объявления типа
Integer	%
Long	&
Single	!
Double	#
Currency	@
String	$

Этот метод типизации данных – пережиток старых версий BASIC; эффективнее объявлять переменные с помощью остальных методов, описанных в настоящей главе.

Если переменная объявлена как локальная, другие процедуры в том же модуле могут использовать подобное имя, но каждый экземпляр переменной считается уникальным в своей процедуре.

Как правило, локальные переменные – самые эффективные, так как VBA освобождает память, которую они используют, после окончания выполнения процедуры.

Переменные уровня модуля. Иногда необходимо, чтобы переменная была доступна во всех процедурах модуля. В таком случае она объявляется перед первой процедурой модуля (за пределами процедур или функций).

В приведенном ниже примере оператор `Dim` – первая инструкция в модуле. Обе процедуры `MySub` и `YourSub` имеют доступ к переменной `CurrentValue`.

```
Dim CurrentValue As Integer

   Sub MySub()
   ' -[Здесь вводится текст процедуры] -
   End Sub
Sub YourSub()
   ' -[Здесь вводится текст процедуры] -
End Sub
```

Значение переменной уровня модуля не изменяется к окончанию выполнения процедуры.

Переменные `Public`. Чтобы сделать переменную доступной во всех процедурах всех модулей VBA проекта, необходимо объявить переменную на уровне модуля с помощью ключевого слова `Public`, а не `Dim`.

```
Public CurrentRate as Long
```

Ключевое слово `Public` делает переменную `CurrentRate` доступной для любой процедуры проекта, даже для процедур, которые располагаются в других модулях проекта. Этот оператор следует вставить перед первой процедурой модуля. Более того, подобный код объявления переменных должен вводиться в стандартном модуле VBA, а не в коде модуля листа или формы.

Переменные `Static`. Переменные `Static` объявляются на уровне процедуры и сохраняют свое значение после окончания процедуры.

```
Sub MySub()
   Static Counter As Integer
   ' -[Здесь вводится текст процедуры] -
End Sub
```

6.1.5. Работа с константами

Значение переменной может изменяться при выполнении процедуры. Иногда же необходимо использовать именованное значение или строку, которая никогда не меняется – *константу*.

Объявление констант. Константы объявляются с помощью оператора `Const`. Ниже приведено несколько примеров.

```
Const NumQuarters as Integer = 4
   Const Rate = .0725, Period = 12
   Const ModName as String = "Budget Macros"
Public Const AppName as String = "Budget Application"
```

Во втором примере тип данных не объявлен. Следовательно, указанные две константы имеют тип `Variant`. Так как константы никогда не изменяют своего значения, обычно их объявляют в виде конкретного типа данных.

Как и переменные, константы также имеют область действия. Если требуется, чтобы константа была доступна только в одной процедуре, ее следует объявить после оператора `Sub` или `Function` – она станет *локальной*. Константа будет доступной для всех процедур в модуле, если ее объявить перед первой процедурой модуля. Чтобы сделать константу доступной для всех модулей рабочей книги, следует использовать ключевое слово `Public` и объявить константу перед первой процедурой модуля.

```
Public Const InterestRate As Double = 0.0725
```

При попытке изменить значение константы в процедуре VBA будет получена ошибка, т. к. константа – это *постоянное значение*, а не *переменная*.

Использование предопределенных констант. В Excel и VBA существует целый ряд предопределенных констант, которые можно использовать без объявления. В этом случае не обязательно знать значение этих констант для их применения. При записи макросов обычно используются константы, а не значения. В следующей процедуре для изменения ориентации страницы активного листа на альбомную применена встроенная константа (`xlLandscape`).

```
Sub SetToLandscape()
   ActiveSheet.PageSetup.Orientation = xlLandscape
End Sub
```

Константа `xlLandscape` была обнаружена путем записи макроса. Описание константы также приводится в справочной системе.

Фактическое значение переменной `xlLandscape` равно 2. Еще одна встроенная константа для изменения ориентации страницы – `xlPortrait` – имеет значение 1. Очевидно, что при использовании встроенных констант не обязательно знать их значения.

Рекомендации по именованию переменных. Распространенное правило именования связано с использованием стандартной приставки в нижнем регистре в имени переменной. Например, булеву переменную, которая контролирует сохранение рабочей книги, можно назвать `bWasSaved`. Таким образом, понятно, что это переменная типа `Boolean`. В приведенной ниже таблице I.6.4 перечислены стандартные префиксы для некоторых типов данных.

Таблица I.6.4

Стандартные префиксы

Тип данных	Префикс
Boolean	b
Integer	i
Long	l
Single	s
Double	d
Currency	c
Date/Time	dt
String	str
Object	obj
Variant	v
Пользовательский	u

6.1.6. Управление строками

Как и Excel, VBA может работать не только с числами, но и с текстом (строками). В VBA представлено два типа строк:

– *строки фиксированной длины* объявляются с определенным количеством символов. Максимальная длина строки составляет 65535 символов;

– *строки переменной длины* теоретически могут вмещать до 2 млрд символов.

Каждый символ в строке занимает 1 байт памяти, к тому же, память дополнительно используется для хранения заголовка строки. При объявлении строки с помощью оператора `Dim` можно определить ее длину, если она известна (задать тип строки с фиксированной длиной), или активизировать динамическую обработку длины строки (задать тип строки переменной длины). Работа со строками фиксированной длины несколько эффективнее с точки зрения использования памяти.

В следующем примере переменная `MyString` объявляется как строка с максимальной длиной 50 символов. `YourString` тоже объявлена как строка, но она имеет переменную длину.

```
Dim MyString As String * 5 0
Dim YourString As String
```

6.1.7. Работа с датами

Для управления датами используется специальный тип данных `Date`.

Переменная, определенная как `Date`, занимает 8 байт памяти и может содержать даты в диапазоне с 1 января 100 года до 31 декабря 9999 года. Тип данных `Date` также применяется для хранения значений времени. В VBA дата и время определяются как значения, заключенные между знаками #.

Диапазон дат, которые можно обрабатывать в VBA, намного шире, чем собственный диапазон дат Excel, который начинается с 1 января 1900 года. Поэтому надо следить, чтобы в рабочем листе не использовались данные, которые находятся за пределами приемлемого диапазона дат Excel.

Ниже приведены отдельные примеры объявления переменных и констант с типом данных `Date`.

```
Dim Today As Date
   Dim StartTime As Date
   Const FirstDay As Date = #1/1/2001#
Const Noon = #12:00:00#
```

Константы дат всегда определяются с помощью формата месяц/день/год, даже если система установлена на отображение данных в другом формате (например, день/месяц/год).

Если для отображения даты используется окно сообщений, дата будет представлена в соответствии с коротким форматом даты, установленным в системе.

6.1.8. Операторы присвоения

Оператор присвоения – это инструкция VBA, выполняющая математическое вычисление и присваивающая результат переменной или объекту. В справочной системе Excel выражение определяется как комбинация ключевык слов, операторов, переменных и констант. Эта комбинация возвращает в результате строку, число или объект. Выражение может осуществлять вычисление, обрабатывать символы или тестировать данные.

В VBA оператором присвоения выступает знак равенства (=). Ниже приведены примеры использования операторов присвоения (выражения приводятся справа от знака равенства).

```
x = 1
   x = x + 1
   x = (y * 2) / (z * 2)
   FileOpen = True
   FileOpen = Not FileOpen
Range("The Year").Value = 2001
```

Выражения могут быть очень сложными. Чтобы сделать длинные выражения более удобными для восприятия, рекомендуется использовать символ продолжения строки (пробел с подчеркиванием).

Зачастую в выражениях применяются функции. Это могут быть встроенные функции VBA, функции рабочих листов Excel или специальные функции, разработанные в VBA.

В VBA математические операторы играют главную роль. Известные операторы описывают математические операции, в том числе сложение (+), умножение (*), деление (/), вычитание (–), возведение в степень (A) и конкатенацию строк (&). Менее знакомые операторы: обратная косая черта (\) – используется в целочисленном делении, оператор `Abs` – применяется при определении модуля числа. Оператор `Mod` возвращает остаток от деления одного числа на другое. Например, следующее выражение возвращает 2.

```
17 Mod 3
```

VBA поддерживает операторы сравнения, которые применяются в формулах Excel: равно (=), больше (>), меньше (<), больше или равно (>=), меньше или равно (<=) и не равно (<>).

Кроме того, VBA представляет полный набор булевых операторов (табл. I.6.5). Полную информацию о них (с примерами) можно найти в справочной системе по VBA.

Таблица I.6.5

Булевы операторы VBA

Оператор	Действие
`Not`	Логическое отрицание выражения
`And`	Логическая конъюнкция двух выражений
`Or`	Логическая дизъюнкция двух выражений
`XoR`	Логическое отрицание двух выражений
`Eqv`	Логическая эквивалентность двух выражений
`Imp`	Логическая импликация двух выражений

Порядок приоритетности выполнения операторов VBA такой же, как и в Excel. Изменить приоритет операторов по умолчанию можно с помощью скобок.

В следующей инструкции используется оператор `Not` для отображения линий сетки в активном окне. Свойство `DisplayGridLines` принимает значение `True` или `False`. Следовательно, применение оператора `Not` изменяет `True` на `False`, а `False` – на `True`.

```
ActiveWindow.DisplayGridLines = _
     Not ActiveWindow.DisplayGridLines
```

Представленное далее выражение осуществляет логическую операцию `And`. Оператор `MsgBox` отображает `True`, только если `Лист1` – активный лист и активная ячейка находится в строке 1. Если одно или оба этих условия неверны, оператор `MsgBox` отображает `False`.

```
MsgBox ActiveSheet.Name = "Лист1" And ActiveCell.Row = 1
```

Следующее выражение осуществляет логическую операцию `Or`. Оператор `MsgBox` отображает `True`, если активен `Лист1` или `Лист2`.

```
MsgBox ActiveSheet.Name = "Лист1" _
     Or ActiveSheet.Name = "Лист2"
```

6.1.9. Массивы

Массив – это группа элементов одного типа, которые имеют общее имя; на конкретный элемент массива ссылаются, используя имя массива и индекс. Например, можно определить массив из 12-ти строк так, чтобы каждая переменная соответствовала названию месяца. Если массив назвать `MonthNames`, то можно обратиться к первому элементу массива как `MonthNames(0)`, ко второму – как `MonthNames(1)` и т. д., до `MonthNames(11)`.

Объявление массивов. Массив, как и обычные переменные, объявляется с помощью операторов `Dim` или `Public`. Кроме того, можно определить количество элементов в массиве: ввести нижний индекс, ключевое слово `To` и верхний индекс – вся конструкция будет заключена в скобки. Например, объявить массив, содержащий ровно 100 целых чисел можно следующим образом:

```
Dim MyArray(1 To 100) As Integer
```

При объявлении массива обязательно следует указывать только верхний индекс, тогда VBA установит нижний индекс равным нулю. Следовательно, два следующих оператора приведут к одинаковым результатам.

```
Dim MyArray(0 To 100) As Integer
Dim MyArray(100) As Integer
```

В обоих случаях массив состоит из 101-го элемента.

Если нужно, чтобы в качестве первого индекса всех массивов использовалась единица, то следует перед первой процедурой модуля сделать следующее объявление:

```
Option Base 1
```

Объявление многомерных массивов. В примерах массивов в предыдущем разделе использовались одномерные массивы. Массивы VBA могут иметь до 60-ти измерений, хотя на самом деле используется не более трех (трехмерные массивы). Показанный ниже оператор объявляет двухмерный 100-элементный массив целых чисел.

```
Dim MyArray(1 To 10, 1 To 10) As Integer
```

Этот массив можно рассматривать как матрицу значений 10×10. Чтобы обратиться к конкретному элементу двумерного массива, нужно использовать два индекса. Например, таким образом присваивается значение элементу предыдущего массива.

```
MyArray(3, 4) = 125
```

Трехмерный массив можно рассматривать как куб, но не существует способа визуально представить данные массива, в котором больше трех измерений.

Динамический массив не имеет предопределенного количества элементов. Динамический массив объявляется с незаполненными значениями в скобках.

```
Dim MyArray() As Integer
```

Тем не менее, прежде чем динамический массив можно будет использовать в программе, необходимо обратиться к оператору `ReDim`, указывающему VBA сколько элементов находится в массиве (или `ReDim Preserve`, если вы решили сохранить текущую длину массива). Оператор `ReDim` можно использовать сколько угодно раз, изменяя, если требуется, размер массива.

6.1.10. Переменные объектов

Переменная объекта – это переменная, представляющая целый объект, например, диапазон или рабочий лист. Переменные объектов имеют особое значение по следующим причинам:

– значительно упрощают программу;

– ускоряют выполнение программы.

Переменные объектов, как и обычные переменные, объявляются с помощью оператора `Dim` или `Public`. Например, в следующем операторе переменная `InputArea` объявляется как объект `Range`.

```
Public InputArea As Range
```

Чтобы узнать, каким образом переменные объектов упрощают программу, проанализируем процедуру, написанную без их использования.

```
Sub NoObjVar()
   Worksheets("Лист1").Range("A1").Value = 124
   Worksheets("Лист1").Range("A1").Font.Bold = True
   Worksheets("Лист1").Range("A1").Font.Italic = True
End Sub
```

Эта процедура вводит значение в ячейку `A1` листа `Лист1` активной рабочей книги, а затем делает начертание содержимого ячейки полужирным и курсивным. В примере введено достаточно много кода для решения такой простой задачи. Чтобы упростить ее, сведем процедуру к использованию объектной переменной.

```
Sub ObjVar()
   Dim MyCell As Range
   Set MyCell = Worksheets("Лист1").Range("A1")
   MyCell.Value = 124
   MyCell.Font.Bold = True MyCell.Font.Italic = True
End Sub
```

После объявления переменной `MyCell` как объекта `Range` оператор `Set` присваивает ей сам объект. В результате в следующих операторах используется упрощенная ссылка `MyCell` вместо длинной `Worksheets("Лист1").Range("A1")`.

После присвоения переменной объекта VBA получает доступ к нему быстрее, чем с помощью непосредственной ссылки на объект. Поэтому, если важна скорость выполнения операции, следует использовать переменные объектов.

6.1.11. Пользовательские типы данных

VBA позволяет создавать специальные, или пользовательские, типы данных. Определенный пользователем тип данных может облегчить управление некоторыми типами данных. Например, если приложение обрабатывает сведения о клиенте, то можно создать пользовательский тип данных с названием `CustomerInfo`.

```
Type CustomerInfo
   Company As String * 25
   Contact As String * 15
   RegionCode As Integer
   Sales As Long
End Type
```

Пользовательские типы данных определяются вверху модуля перед началом введения кода процедур.

Если пользовательский тип данных уже создан, для объявления переменной этого типа применяется оператор `Dim`. Обычно пользовательский тип данных определяется для массивов.

```
Dim Customers(1 To 100) As CustomerInfo
```

Все 100 элементов этого массива состоят из четырех компонентов (как указано в пользовательском типе данных – `CustomerInfo`). На конкретный компонент элемента можно сослаться следующим образом:

```
Customers(1).Company = "Acme Tools"
   Customers(1).Contact = "Tim Robertson"
   Customers(1).RegionCode = 3
Customers(1).Sales = 150677
```

Также существует возможность управлять элементом массива как одним целым. Например, чтобы скопировать информацию из `Customers(1)` в `Customers(2)`, используется следующая инструкция:

```
Customers(2) = Customers(1)
```

Предыдущий пример эквивалентен приведенному ниже блоку инструкций.

```
Customers(2).Company = Customers(1).Company
   Customers(2).Contact = Customers(1).Contact
   Customers(2).RegionCode = Customers(1).RegionCode
Customers(2).Sales = Customers(1).Sales
```

6.1.12. Встроенные функции

Как и в большинстве других языков программирования, в VBA есть ряд встроенных функций, упрощающих вычисления и операции. Часто функции позволяют выполнить операции, которые по-другому осуществить сложно или вообще невозможно. Многие функции VBA подобны (или идентичны) функциям Excel. Подробное описание всех функций можно найти в справочной системе.

Чтобы получить список функций VBA при написании кода, следует ввести VBA и точку (.). VBE отображает список всех вложенных в объект VBA объектов, включая функции (рис. I.6.2). Функции обозначаются зеленым значком.

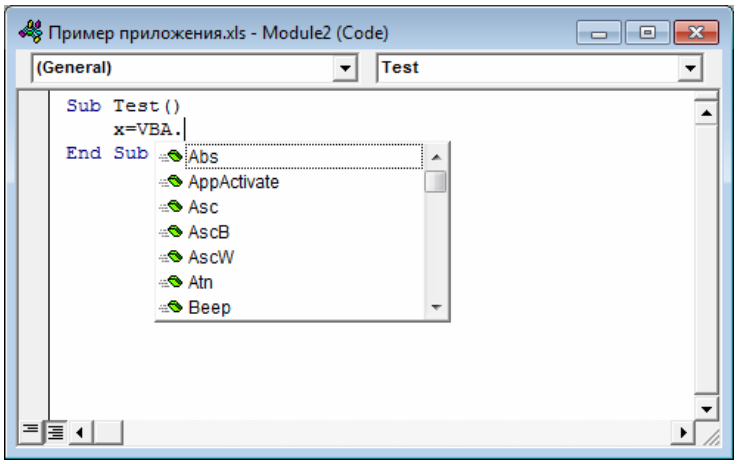


Рис. I.6.2. Отображение списка функций VBA в VBE

Функции VBA используются в выражениях почти так же, как и функции Excel в формулах рабочего листа. Например, можно создавать вложенные функции VBA.

Ниже приведена простая процедура, которая вычисляет квадратный корень переменной с помощью функции Sqr VBA, сохраняет результат в другой переменной и затем отображает результат.

```
Sub ShowRoot()
   MyValue = 25
   SquareRoot = Sqr(MyValue)
   MsgBox SquareRoot
End Sub
```

Функция Sqr VBA эквивалентна функции рабочего листа КОРЕНЬ в Excel. Можно использовать некоторые (но не все) функций Excel в коде VBA. Объект WorksheetFunction, который содержится в объекте Application, располагает всеми функциями рабочего листа, которые можно вызвать в процедуре VBA.

Чтобы использовать функцию Excel в операторе VBA, перед названием функции введите следующее выражение:

```
Application.WorksheetFunction
```

Приведенный ниже пример показывает, как используется функция Excel в процедуре VBA. Нечасто используемая функция Excel ROMAN преобразует десятичное число в римское.

```
Sub ShowRoman()
   DecValue = 2001
   RomanValue = Application.WorksheetFunction.Roman(DecValue)
   MsgBox RomanValue
End Sub
```

6.1.13. Управление объектами и коллекциями

VBA предлагает две конструкции, которые упрощают управление объектами и коллекциями:

– конструкция With-End With;

– конструкция For Each-Next.

Конструкция With-End With. Конструкция With-End With позволяет выполнять несколько операций над одним объектом. Чтобы понять, как она работает, проанализируем следующую процедуру, которая изменяет пять свойств выделенного объекта (подразумевается, что выделен объект Range (одна ячейка или диапазон ячеек)).

```
Sub ChangeFont1()
   Selection.Font.Name = "Times New Roman"
   Selection.Font.FontStyle = "Bold Italic"
   Selection.Font.Size = 12
   Selection.Font.Underline = xlUnderlineStyleSingle
   Selection.Font.ColorIndex = 5
End Sub
```

Эту процедуру можно переписать с помощью конструкции `With-End With`. Процедура, показанная ниже, работает точно так же, как и предыдущая.

```
Sub ChangeFont2()
   With Selection.Font
          .Name = "Times New Roman" .FontStyle = "Bold Italic"
          .Size = 12
          .Underline = xlUnderlineStyleSingle
          .ColorIndex = 5
   End With
End Sub
```

На первый взгляд, второй вариант этой процедуры читать сложнее. Однако надо учитывать, что целью изменений является увеличение скорости выполнения операций. Первый вариант более прямолинейный и его легче понять, но процедура, использующая для изменения нескольких свойств одного объекта конструкцию `With-End With`, помогает повысить эффективность выполнения кода по сравнению с эквивалентной ей процедурой, которая явно ссылается на объект в каждом операторе.

Конструкция `For Each-Next`. Итак, нам уже известно, что коллекция – это группа однородных объектов (см. раздел 5.1.1). Например, коллекция `Workbooks` – это коллекция всех открытых рабочих книг `Workbook`. Существует ряд других коллекций, с которыми можно работать. Чтобы правильно использовать конструкцию `For Each-Next`, необязательно знать, сколько элементов насчитывает коллекция.

Предположим, нужно выполнить действие над всеми объектами коллекции или необходимо оценить все объекты коллекции и совершить действие при выполнении определенных условий. Это идеальная ситуация для применения конструкции `For Each-Next`.

Синтаксис конструкции `For Each-Next` приведен ниже:

```
For Each элемент In группа
   [инструкции]
   [Exit For]
   [инструкции]
Next [элемент]
```

Следующая процедура использует конструкцию For Each-Next для обращения поочередно к каждому из шести элементов массива фиксированной длины.

```
Sub Macro1()
   Dim MyArray(5)
   For i = 0 To 5
          MyArray(i) = Rnd
   Next i
   For Each n In MyArray
          Debug.Print n
   Next n
End Sub
```

В процедуре, приведенной далее, применена конструкция For Each-Next по отношению к коллекции Sheets активной рабочей книги. При выполнении этой процедуры функция MsgBox отображает свойство Name каждого рабочего листа. (Если в активной рабочей книге пять листов, функция MsgBox вызывается пять раз).

```
Sub CountSheets()
   Dim Item as WorkSheet
   For Each Item In ActiveWorkBook.WorkSheets
          MsgBox Item.Name
   Next Item
End Sub
```

В предыдущем примере Item – это переменная объекта (точнее, объект Worksheet). В названии Item не заключен особый смысл; вместо него можно использовать любое корректное название переменной.

В следующем примере конструкция For Each-Next используется для циклического просмотра всех объектов коллекции Windows.

```
Sub HiddenWindows()
   Dim AllVisible As Boolean
   Dim Item As Window
   AllVisible = True
   For Each Item In Windows
          If Item.Visible = False Then
               AllVisible = False
               Exit For
               End If
          Next Item
          MsgBox AllVisible
End Sub
```

Если окно скрыто, то значение переменной `AllVisible` изменяется на `False` и процедура выкодит из цикла `For Each-Next`. В окне сообщения отображается `True` (если все окна видимы) и `False` (если хотя бы одно окно скрыто). Оператор `Exit For` необязателен. Он предоставляет способ досрочно выйти из цикла `For Each-Next`. Обычно заданная конструкция применяется вместе с оператором `If-Then` (см. далее в этой главе).

Ниже приведен пример закрытия всех рабочих книг, кроме активной. Процедура использует конструкцию `If-Then` для оценки каждой рабочей книги в коллекции `Workbooks`.

```
Sub CloseInActive()
   Dim Book as Workbook
   For Each Book In Workbooks
          If Book.Name <> ActiveWorkbook.Name Then Book.Close
          Next Book
End Sub
```

Последний представленный в этом разделе пример `For Each-Next` выполняется после того, как пользователь выделит диапазон ячеек. В данном случае объект `Selection` играет роль коллекции, состоящей из объектов `Range`, так как каждая ячейка в выщеленном диапазоне представляет собой объект `Range`. Процедура оценивает каждую ячейку и использует функцию VBA `UCase` для преобразования текста, располагающегося в ней, в верхний регистр (ячейки с числовыми данными остаются неизмененными).

```
Sub MakeUpperCase()
   Dim Cell as Range
   For Each Cell In Selection
          Cell.Value = UCase(Cell.Value)
   Next Cell
End Sub
```

6.1.14. Контроль выполнения кода

Некоторые процедуры VBA начинают выполняться с первых строк кода. Этот процесс производится построчно до самого конца процедуры (например, так всегда работают макросы, записанные при выполнении действий). Однако порой необходимо контролировать последовательность операций, пропуская отдельные операторы, повторно выполняя некоторые команды и проверяя условия для определения следующего действия, выполняемого процедурой.

В предыщущем разделе описана конструкция `For Each-Next`, которая является циклической структурой. В настоящем разделе речь пойдет о дополнительных способах контроля выполнения процедур, написанных на VBA:

– операторы `GoTo`;
– конструкции `If-Then`;
– конструкции `Select Case`;
– циклы `For-Next`;
– циклы `Do While`;
– циклы `Do Until`.

Операторы `GoTo`. Самый простой способ изменить последовательность операций в коде – использовать оператор `GoTo`. Он перенаправляет ход выполнения программы на новую инструкцию, которая помечена специальным образом (текстовая строка, заканчивающаяся двоеточием, или число без двоеточия, указанные перед инструкцией). Процедуры VBA могут содержать любое количество меток, а оператор `GoTo` не определяет переход за пределы процедуры.

В приведенной ниже процедуре применена функция VBA `InputBox` для получения имени пользователя. Если имя пользователя отличается от *Ховард*, то процедура переходит к метке `WrongName`, на чем заканчивает свою работу. В противном случае процедура выполняет дополнительные операции. Оператор `Exit Sub` заканчивает выполнение процедуры.

```
Sub GoToDemo()
   UserName = InputBox("Введите свое имя:")
   If UserName <> "Ховард" Then GoTo WrongName
   MsgBox ("Привет, Ховард.")
   '   - [Здесь вводится дополнительный код]
   Exit Sub
   WrongName:
   MsgBox "Извините, эту процедуру может запускать только Ховард."
End Sub
```

Представленная процедура работает, но оператор `GoTo`, как правило, используется, если другого способа выполнить действие просто не существует. Единственной ситуацией, когда оператор `GoTo` в VBA действительно необходим, является перехват ошибок.

Конструкция `If-Then`. Конструкция `If-Then` является одной из наиболее часто используемых для группировки инструкций VBA. Эта по-

пулярная конструкция наделяет приложения способностью принимать решения. Такая способность является ключевой при создании эффективных программ. Стандартный синтаксис конструкции If-Then таков:

```
If условие Then инструкции_истина [Else инструкции_ложь]
```

Конструкция If-Then используется для выполнения одного или более операторов при справедливости заданного условия. Оператор Else необязателен. Он позволяет выполнять одну или более инструкций в случае несправедливости условия.

В описанной ниже процедуре применена структура If-Then без оператора Else. Пример связан с управлением временными данными. VBA использует систему дат и времени, похожую на задействованную в Excel. Время дня выражается дробным числом – например, полдень представлен как 0.5. Функция VBA Time возвращает значение, представляющее время в формате системных часов. В следующем примере сообщение отображается, если текущее время меньше полудня. Если текущее системное время больше или равно 0.5, то процедура заканчивается, и ничего не происходит.

```
Sub GreetMe1()
   If Time < 0.5 Then MsgBox "Доброе утро"
End Sub
```

Можно отобразить другое приветствие, если текущее время больше полудня. Для этого надо добавить еще один оператор If-Then.

```
Sub GreetMe2()
   If Time < 0.5 Then MsgBox "Доброе утро"
   If Time >= 0.5 Then MsgBox "Добрый день"
End Sub
```

Обратите внимание, что для второй конструкции If-Then использован оператор >= (больше или равно), таким образом учитывается тот редкий случай, когда время на часах точно равно 12.00.

Другой подход – использовать оператор Else конструкции If-Then.

```
Sub GreetMe3()
   If Time < 0.5 Then MsgBox "Доброе утро" Else _
          MsgBox "Добрый день"
End Sub
```

В данном случае введен символ продолжения строки, а `If-Then-Else` является одним оператором.

Если необходимо расширить процедуру до обработки трех условий (например, утро, день и вечер), то можно использовать либо три оператора `If-Then`, либо вложенную структуру `If-Then-Else`. Первый вариант значительно проще.

```
Sub GreetMe4()
   If Time < 0.5 Then MsgBox "Доброе утро"
   If Time >= 0.5 And Time < 0.75 Then MsgBox "Добрый день"
   If Time >= 0.75 Then MsgBox "Добрый вечер"
End Sub
```

Значение `0.75` представляет время `18:00` – три четверти суток и тот момент, когда день переходит в вечер.

В предыдущих примерах каждая инструкция в процедуре выполняется, даже утром. Для эффективности следует включить структуру, заканчивающую процедуру, когда одно из условий выполняется. Если, например, отображается сообщение `Доброе утро`, то процедура заканчивается без проверки других, излишних условий. Конечно, разница в скорости выполнения обеих процедур несущественна, если разрабатывается такая маленькая процедура, как рассматриваемая. Но в более сложных приложениях все же рекомендуется учесть следующий синтаксис:

```
If условие Then
   [операторы_истина]
   [Elseif условие-n Then
   [альтернативные_операторы]]
   [Else
   [операторы_по_умолчанию]]
End If
```

Этот же синтаксис можно использовать для ввода кода процедуры `GreetMe`.

```
Sub GreetMe5()
   If Time < 0.5 Then
          MsgBox "Доброе утро"
   Elseif Time >=0.5 And Time < 0.75 Then
          MsgBox "Добрый день"
          Else
          MsgBox "Добрый вечер"
          End If
End Sub
```

Если в представленном синтаксисе условие выполняется, то выполняются соответствующие операторы, после чего конструкция `If-Then` заканчивается. Другими словами, при этом не оцениваются дополнительные условия. Таким образом, этот синтаксис наиболее эффективен.

Далее представлен еще один способ реализовать рассматриваемый пример. В нем используются вложенные конструкции `If-Then-Else` (без `ElseIf`). Данная процедура также эффективна, и ее легко понять. Здесь для каждого оператора `If` существует свой оператор `End If`.

```
Sub GreetMe6()
          If Time < 0.5 Then
          MsgBox "Доброе утро"
          Else
          If Time >=0.5 And Time < 0.75 Then
               MsgBox "Добрый день"
          Else
               If Time >=0.75 Then
                    MsgBox "Добрый вечер"
               End If
          End If
   End If
End Sub
```

Ниже продемонстрирован еще один пример, использующий простую форму конструкции `If-Then`. Процедура запрашивает у пользователя значение переменной `Quantity` и отображает скидку на основе полученного значения. Если в окне `InputBox` пользователь щелкнет на кнопке Отмена, то переменная `Quantity` приравняется пустой строке, в результате процедура заканчивается. Следует отметить, что эта процедура не выполняет других проверок (например, мы не проверяем в данном случае введенное числовое значение на отрицательность).

```
Sub Discount1()
   Quantity = InputBox("Введите значение: ")
          If Quantity = "" Then Exit Sub
          If Quantity >= 0 Then Discount = 0.1
          If Quantity >= 25 Then Discount = 0.15
          If Quantity >= 50 Then Discount = 0.2
          If Quantity >= 75 Then Discount = 0.25
          MsgBox "Скидка: " & Discount
End Sub
```

Обратите внимание, что каждый оператор If-Then в представленной процедуре всегда выполняется, а значение Discount может изменяться. Однако в результате всегда отображается нужное значение.

Следующая процедура – это вариант предыдущей, переписанной с использованием альтернативного синтаксиса. Процедура будет окончена после выполнения блока *операторы_истина*.

```
Sub Discount2()
          Quantity = InputBox("Введите значение: ")
          If Quantity = "" Then Exit Sub
          If Quantity >= 0 And Quantity < 25 Then
               Discount = 0.1
          ElseIf Quantity < 50 Then
               Discount = 0.15
          ElseIf Quantity < 75 Then
               Discount = 0.2
          ElseIf Quantity >= 75 Then
               Discount = 0.25
          End If
               MsgBox "Скидка: " & Discount
End Sub
```

Вложенные структуры If-Then достаточно громоздкие. Поэтому рекомендуется использовать их только для принятия простых бинарных решений. Если же необходимо выбрать между тремя и более вариантами, то целесообразно обратиться к конструкции Select Case.

Конструкции Select Case. Конструкция Select Case применяется при выборе между тремя и более вариантами. Она справедлива также для двух вариантов и является хорошей альтернативой структуре If-Then-Else. Конструкция Select Case имеет следующий синтаксис:

```
Select Case тестируемое_выражение
          [Case список условий-n
          [операторы-n]]
          [Case Else
          [операторы_по_умолчанию]]
End Select
```

Приведем пример конструкции Select Case, который показывает еще один способ запрограммировать процедуру GreetMe, рассмотренную ранее.

```
Sub GreetMe()
   Select Case Time
          Case Is < 0.5
               Msg = "Доброе утро"
          Case 0.5 To 0.75
               Msg = "Добрый день"
          Case Else
               Msg = "Добрый вечер"
   End Select
          MsgBox Msg
End Sub
```

Ниже приведена версия процедуры `Discount`, переписанная с использованием конструкции `Select Case`. Эта процедура предполагает, что `Quantity` всегда выражается целым числом. Чтобы упростить процедуру, в ней не выполняется проверка введенных данных.

```
Sub Discount3()
          Quantity = InputBox("Введите значение: ")
   Select Case Quantity
          Case ""
               Exit Sub
               Case 0 To 24
               Discount = 0.1
               Case 25 To 49
               Discount = 0.15
          Case 50 To 74
               Discount = 0.2
          Case Is >= 75
               Discount = 0.25
          End Select
          MsgBox "Скидка: " & Discount
End Sub
```

В операторе `Case` может также использоваться оператор `Or`. В следующей процедуре для определения, каким днем является текущий (субботой или воскресеньем), применена функция VBA `WeekDay`, после чего отображается соответствующее сообщение. Обратите внимание на использование оператора `Or`, который проверяет день недели на принадлежность к субботе или воскресенью.

```
Sub GreetUser()
          Select Case Weekday(Now)
          Case 1 Or 7
          MsgBox "Это выходные."
          Case Else
          MsgBox "Это не выходные."
          End Select
End Sub
```

Под каждым оператором Case может указываться любое количество инструкций, и они выполняются при условии Case, имеющем значение True. Если вы используете в каждом случае Case только одну инструкцию (как в предыщущем примере), то можете поместить ее в той же строке, что и ключевое слово Case (но не забудьте про символ разделения операторов в VBA – двоеточие). Этот прием сделает текст программы более компактным.

```
Sub Discount3()
          Quantity = InputBox("ВВеflНТе количество: ")
          Select Case Quantity
               Case "": Exit Sub
               Case 0 To 24: Discount = 0.1
               Case 25 To 49: Discount = 0.15
               Case 50 To 74: Discount = 0.2
               Case Is >= 75: Discount = 0.25
          End Select
          MsgBox "Скидка: " & Discount
End Sub
```

VBA выходит из конструкции Select Case, как только найдено условие Case, которое имеет значение True. Следовательно, для максимальной эффективности в первую очередь следует выполнять проверку наиболее вероятного случая.

Структуры Select Case можно вкладывать друг в друга. Например, следующая процедура выполняет проверку состояния окна Excel (развернутое, свернутое, нормальное), а затем отображает сообщение с описанием состояния. Если окно Excel находится в нормальном состоянии, то процедура проверяет состояние активного окна и отображает еще одно сообщение.

```
Sub AppWindow()
          Select Case Application.WindowState
               Case xlMaximized: MsgBox "Окно приложения раз-
               вернуто"
               Case xlMinimized: MsgBox "Окно приложения свер-
               нуто"
               Case xlNormal: MsgBox "Окно приложения в нормальном со-
               стоянии"
          Select Case ActiveWindow.WindowState
               Case xlMaximized: MsgBox "Окно книги развернуто"
               Case xlMinimized: MsgBox "Окно книги свернуто"
               Case xlNormal: MsgBox "Окно книги в нормальном состоянии"
          End Select
        End Select
     End Sub
```

Можно создавать конструкции `Select Case` любой степени вложенности, но следует убедиться, что каждому оператору `Select Case` соответствует свой оператор `End Select`.

Циклическая обработка инструкций. Алгоритм *циклической* структуры предусматривает многократное повторение действий в одной и той же последовательности по одним и тем же математическим зависимостям, но при разных значениях некоторой специально изменяемой величины. Циклические алгоритмы позволяют существенно сократить объем программы за счет многократного выполнения группы повторяющихся вычислений, так называемого *тела цикла*.

Специально изменяемый по заданному закону параметр, входящий в тело цикла, называется *переменной цикла*. Переменная цикла используется для подготовки очередного повторения цикла и отслеживания условий его окончания. В качестве переменной цикла используют любые переменные, индексы массивов, аргументы вычисляемых функций и тому подобные величины. Во время выполнения тела цикла параметры переменной цикла изменяются в интервале от начального до конечного значения с заданным шагом. Следовательно, при организации циклических вычислений необходимо предусмотреть задание начального значения переменной цикла, закон ее изменения перед каждым новым повторением и ее конечное значение, при достижении которого произойдет завершение цикла.

Циклические алгоритмы разделяют на *детерминированные* и *итерационные*.

Циклы, в которых число повторений заранее известно из исходных данных или определено в ходе решения задачи, называют *детерминированными*.

Циклы, в которых число повторений неизвестно из исходных данных и не определено по ходу решения задачи, называют *итерационными*. В итерационных циклах для организации выхода из тела цикла предусматривается проверка некоторого заранее заданного условия, для чего используют блок проверки условия.

Следующий код, в котором в диапазон вводятся последовательные числа, является примером того, что называют плохим циклом. Процедура использует две переменные для хранения начального значения (`StartVal`) и общего количества ячеек, которые необходимо заполнить (`NumToFill`). В этом цикле для управления порядком выполнения операций используется оператор `GoTo`. Если переменная `Cnt`, отвечающая за то, сколько ячеек заполнено, меньше числа, заданного пользователем, то выполняется переход назад к метке `DoAnother`.

```
Sub BadLoop()
   StartVal = 1
   NumToFill = 100
   ActiveCell.Value = StartVal
   Cnt = 1
DoAnother:
   ActiveCell.Offset(Cnt,0).Value = StartVal + Cnt
          Cnt = Cnt + 1
          If Cnt < NumToFill Then GoTo DoAnother Else Exit Sub
End Sub
```

Описанная процедура выполняется правильно. Почему же она является примером плохого цикла? Квалифицированные программисты не используют оператор GoTo, если можно обойтись без него. Этот оператор значительно усложняет восприятие кода, поскольку в данном случае практически невозможно структурировать цикл с помощью отступов. Кроме того, такой тип неструктурированного цикла делает процедуру подверженной частым ошибкам. Также использование большого количества меток приводит к получению программы с плохой структурой (или без структуры вообще) и бессистемным порядком следования операций.

Поскольку в VBA встроено несколько структурированных команд циклов, в процессе принятия решений практически никогда не требуется прибегать к операторам GoTo.

ЦИКЛЫ FOR-NEXT

Простейший пример хорошего цикла – For-Next. Этот оператор имеет следующий синтаксис:

```
For счетчик = начало To конец [Step шаг]
          [инструкции]
          [Exit For]
          [инструкции]
Next [счетчик]
```

Ниже приведен пример цикла For-Next, в котором не используется переменная шаг и необязательный оператор Exit For. Эта процедура выполняет оператор Sum = Sum + Sqr(Count) 100 раз и отображает результат – сумму квадратных корней первых 100 целых чисел.

```
Sub SumSquareRoots()
   Dim Sum As Double, Count As Integer
          Sum = 0
          For Count = 1 To 100
   Sum = Sum + Sqr(Count)
   Next Count
          MsgBox Sum
End Sub
```

В данном примере Count (переменная-счетчик цикла) имеет начальное значение 1 и увеличивается на 1 при каждом повторении цикла. Переменная Sum суммирует квадратные корни каждого значения Count.

Используя циклы For-Next, важно понимать, что счетчик цикла является обычной переменной. В результате значение счетчика цикла можно изменять в блоке программы, который выполняется между операторами For и Next. Однако это очень неудачный прием, так как он может вызвать непредсказуемые результаты. Необходимо принять специальные меры предосторожности, чтобы удостовериться, что программа не изменяет счетчик цикла.

Можно также использовать переменную шаг, чтобы пропустить отдельные итерации цикла. Ниже рассмотрена исходная процедура, переписанная так, что суммируются квадратные корни всех нечетных чисел в диапазоне от 1 до 100.

```
Sub SumOddSquareRoots() Sum = 0
          For Count = 1 To 100 Step 2
   Sum = Sum + Sqr(Count)
          Next Count
          MsgBox Sum
End Sub
```

В этой процедуре исходное значение Count равно 1, затем счетчик принимает значения 3, 5, 7 и т. д. Последнее значение Count, используемое в цикле, равно 99. Когда цикл заканчивается, значение Count равно 101.

Представленная далее процедура выполняет ту же задачу, что и BadLoop, описанный в начале раздела «Циклическая обработка инструкций». Однако в данном случае не используется оператор GoTo, поэтому неудачный цикл превращается в удачный. В нем задействована структура For-Next.

```
Sub GoodLoop()
   StartVal = 1
   NumToFill = 100
   ActiveCell.Value = StartVal
          For Cnt = 0 To NumToFill - 1
   ActiveCell.Offset(Cnt,0).Value = StartVal + Cnt
          Next Cnt
End Sub
```

Циклы For-Next могут также содержать один или более операторов Exit For. Когда программа встречает этот оператор, то сразу же выходит из цикла, как показано в следующем примере. Эта процедура определяет, какая ячейка столбца A на активном рабочем листе имеет наибольшее значение.

```
Sub ExitForDemo()
          MaxVal = Applica-
          tion.WorksheetFunction.Max(Range("A:A"))
          For Row = 1 To 65536
               Set TheCell = Range("A1").Offset(Row - 1, 0)
               If TheCell.Value = MaxVal Then
          MsgBox "Максимальное значение находится в строке " & Row
          TheCell.Activate
          Exit For
   End If
          Next Row
End Sub
```

Максимальное значение в столбце вычисляется с помощью функции Excel MAX. Затем это значение присваивается переменной MaxVal. Цикл For-Next проверяет каждую ячейку в столбце. Если определенная ячейка равна MaxVal, оператор Exit For заканчивает процедуру. Однако перед выходом из цикла процедура сообщает пользователю о расположении искомой ячейки и активизирует ее.

Пример ExitForDemo представлен с целью продемонстрировать выход из цикла For-Next. Однако это не самый эффективный способ найти максимальное значение в диапазоне. Поставленную задачу может выполнить единственный оператор.

```
Range("A:A").Find(Application.WorksheetFunction.Max _
(Range("A:A"))).Activate
```

В предыдущих примерах использовались достаточно простые циклы. Но в цикл можно помещать любое количество операторов и даже вкладывать циклы For-Next в другие циклы For-Next. Ниже приведен пример, в котором применены вложенные циклы For-Next для инициализации массива 10×10×10 значением -1. По завершении выполнения процедуры все 1000 элементов массива MyArray будут содержать значение -1.

```
Sub NestedLoops()
          Dim MyArray(1 To 10, 1 To 10, 1 To 10)
          Dim i As Integer, j As Integer, k As Integer
          For i = 1 To 10
   For j = 1 To 10
          For k = 1 To 10
               MyArray(i, j, k) = -1
               Next k
          Next j
   Next i
End Sub
```

ЦИКЛЫ DO WHILE

Оператор `Do While` – еще один тип циклической структуры, представленной в VBA. В отличие от цикла `For-Next`, цикл `Do While` выполняется до тех пор, пока выполняется заданное условие. Цикл `Do While` может иметь один из двух представленных ниже синтаксисов:

```
Do [While условие]
   [инструкции]
   [Exit Do]
   [инструкции]
Loop
```

или

```
Do
   [инструкции]
   [Exit Do]
   [инструкции]
Loop [While условие]
```

Как видно, VBA позволяет проверять условие `While` в начале или в конце цикла. Разница между этими двумя синтаксисами связана с моментом, когда оценивается условие. В первом синтаксисе содержимое цикла может вообще не выполняться. Во втором содержимое цикла всегда выполняется (как минимум один раз).

Следующий пример демонстрирует цикл `Do While` с первым синтаксисом.

```
Sub DoWhileDemo()
   Do While Not IsEmpty(ActiveCell)
          ActiveCell.Value = 0
          ActiveCell.OffSet(1, 0).Select
   Loop
End Sub
```

Данная процедура использует активную ячейку как точку отсчета и просматривает значения вниз по столбцу, вставляя ноль в активную ячейку. При каждом повторении цикла активной становится следующая ячейка в столбце. Цикл продолжается, пока функция VBA `IsEmpty` не определит, что активная ячейка пуста.

Далее рассмотрим работу второго варианта синтаксиса цикла `Do While`. Цикл всегда выполняется хотя бы один раз, даже если исходная активная ячейка пуста.

```
Sub DoWhileDemo2()
   Do
          ActiveCell.Value = 0
          ActiveCell.OffSet(1, 0).Select
   Loop While Not IsEmpty(ActiveCell)
End Sub
```

Ниже следует еще один пример цикла `Do While`. Эта процедура открывает текстовый файл, считывает каждую строку, преобразует текст в верхний регистр, а затем сохраняет его на активном листе, начиная с ячейки `A1`, и продолжает перемещаться вниз по столбцу. Представленная процедура использует функцию VBA `EOF`, возвращающую `True`, если достигнут конец файла. Последний оператор закрывает текстовый файл.

```
Sub DoWhileDemo1()
   Open "c:\data\textfile.txt" For Input As #1
   LineCt = 0
   Do While Not EOF(1)
          Line Input #1, LineOfText
          Range("A1").Offset(LineCt, 0) = UCase(LineOfText)
          LineCt = LineCt + 1
   Loop Close #1
End Sub
```

ЦИКЛЫ `DO UNTIL`

Структура цикла `Do Until` имеет много общего с конструкцией `Do While`. Разница заключается в том, как проверяется условие цикла. В варианте `Do While` цикл выполняется до тех пор, пока выполняется условие. В цикле `Do Until` цикл выполняется, пока условие не начнет выполняться.

Структура `Do Until` может быть представлена двумя вариантами синтаксиса.

```
Do [Until условие]
   [инструкции]
   [Exit Do]
   [инструкции]
Loop
```

или

```
Do
   [инструкции]
   [Exit Do]
   [инструкции]
Loop [Until условие]
```

Пример, приводимый далее, уже был продемонстрирован для цикла `Do While`, но теперь он изменен для иллюстрации возможностей цикла `Do Until`. Единственное отличие – строка с оператором `Do`. Этот пример делает программу несколько понятнее, так как не используется отрицание, необходимое в примере `Do While`.

```
Sub DoUntilDemo1()
   Open "c:\data\textfile.txt" For Input As #1
   LineCt = 0
   Do Until EOF(1)
          Line Input #1, LineOfText
          Range("A1").Offset(LineCt, 0) = UCase(LineOfText)
          LineCt = LineCt + 1
   Loop
   Close #1
End Sub
```

6.2. Работа с процедурами VBA

6.2.1. О процедурах

Процедура – это последовательность операторов VBA, расположенная в модуле VBA, доступ к которому обеспечивается с помощью VBE. Модуль может включать любое количество процедур.

Существует несколько способов вызвать, или выполнить, процедуры. Процедура выполняется от начала до конца (этот процесс также можно преждевременно прервать).

Некоторые процедуры получают *аргументы*. *Аргумент* – это информация, используемая процедурой в процессе выполнения. Аргументы процедуры во многом подобны аргументам, используемым функциями Excel. Инструкции в процедуре обычно выполняют логические операции над аргументами, а результаты процедуры обычно основаны на предоставляемых ей аргументах.

6.2.2. Объявление процедуры

При объявлении процедуры с использованием ключевого слова Sub необходимо придерживаться следующего синтаксиса:

```
[Private | Public][Static] Sub имя ([список_аргументов])
   [инструкции]
   [Exit Sub]
   [инструкции]
End Sub
```

Здесь:

– Private (необязательное ключевое слово) – указывает на то, что процедура доступна только для других процедур в том же модуле;

– Public (необязательное ключевое слово) – указывает на то, что процедура доступна для всех остальных процедур во всех модулях рабочей книги;

– Static (необязательное ключевое слово) – указывает на то, что переменные процедуры сохраняются после окончания процедуры;

– Sub (обязательное ключевое слово) – обозначает начало процедуры;

– *имя* – любое корректное название процедуры;

– *список_аргументов* – представляет заключенный в скобки список переменных, содержащих аргументы, которые передаются в процедуру. Для разделения аргументов используется запятая. Если процедура не использует аргументы, то необходимо включить в объявление процедуры пустые скобки;

– *инструкции* (необязательные) – корректные инструкции VBA;

– Exit Sub (необязательный оператор) – вызывает немедленный выход из процедуры до ее формального завершения;

– End Sub (обязательный оператор) – указывает на окончание процедуры.

За некоторым исключением, все инструкции VBA в модуле должны содержаться в процедурах. Эти исключения касаются объявления переменных уровня модуля, определения пользовательских типов данных и некоторых других инструкций, определяющих параметры на уровне модуля.

Название процедуры. Каждая процедура должна иметь название. Правила именования процедур, в основном такие же, как при именовании переменных. В идеале название процедуры должно описывать, что выполняют операторы, содержащиеся в процедуре. Хорошее правило – использование названия, включающего глагол и существительное (например, ProcessDate – обработать данные, PrintReport – распечатать отчет,

`Sort_Array` – сортировать массив или `CheckFilename` – проверить имя файла). Следует избегать незначимых названий (`Dolt`, `Update`, `Fix` и т. п.).

Область действия процедуры. В предыщущей главе отмечалось, что область действия переменной определяется модулями и процедурами, в которых может использоваться переменная. Аналогичным образом область действия процедуры определяет, какие процедуры могут ее вызывать.

ПРОЦЕДУРЫ ТИПА `PUBLIC`

По умолчанию все процедуры имеют область действия `Public`, т. е. они могут быть вызваны другими процедурами в любом модуле рабочей книги. Ключевое слово `Public` использовать необязательно, но его часто включают для ясности. Обе приведенные ниже процедуры имеют область действия `Public`.

```
Sub First()
   ' ...[код процедуры]..
End Sub

Public Sub Second()
   ' ...[код процедуры]..
End Sub
```

ПРОЦЕДУРЫ ТИПА `PRIVATE`

Процедуры типа `Private` могут быть вызваны другими процедурами в этом же модуле, но не процедурами других модулей.

В следующем примере объявляется процедура типа `Private` с названием `MySub`.

```
Private Sub MySub()
   ' ...[код процедуры]
End Sub
```

Выполнение процедуры. Далее представлены некоторые способы *выполнения*, или вызова, процедуры VBA.

1) команда *меню Run → Run Sub/UserForm* (в VBE). Альтернатива – нажать F5. Excel выполняет процедуру, в которой находится курсор. Этот метод не срабатывает, если процедура имеет один или более аргументов. Эта команда используется преимущественно для тестирования процедуры в процессе ее разработки;

2) из диалогового окна «Макрос» программы Excel (которое можно открыть, выполнив команду *меню Сервис → Макрос → Макросы*);

3) с помощью сочетания клавиши Ctrl и присвоенной процедуре клавиши (если процедуре присвоено сочетание клавиш);

4) щелкнув на кнопке или форме рабочего листа. Для этого кнопке или форме должна быть присвоена процедура;

5) из другой процедуры. Для этого существует три способа:

– ввести название процедуры и необходимые аргументы этой процедуры (если они есть) через запятую:

```
Sub MySub()
' ...[код]...
   UpdateSheet
' ...[код]...
End Sub

Sub UpdateSheet()
' ...[код]...
End Sub
```

В данном примере процедура `MySub` выполняет некоторые операторы (не показанные в примере), запускает процедуру `UpdateSheet` и затем выполняет остальные операторы,

– ввести ключевое слово `Call`, после него – название процедуры и ее аргументы (если они есть), заключенные в скобки и разделенные запятыми:

```
Sub MySub()
   MonthNum = InputBox("Введите номер месяца: ")
   Call UpdateSheet(MonthNum)
   ' ...[код]...
   End Sub

Sub UpdateSheet(MonthSeq)
   ' ...[код]...
End Sub
```

В этом примере ключевое слово `Call` вызывает процедуру `Update`, имеющую один аргумент; вызывающая процедура передает аргумент в вызываемую процедуру,

– использовать метод `Run` объекта `Application`. Этот метод можно применять для выполнения других процедур VBA. Метод `Run` полезен, когда выполняется процедура, название которой присвоено переменной. Тогда переменную можно передать в метод `Run` как аргумент:

```
Sub MySub()
   MonthNum = InputBox("BBeflHTe номер месяца: ")
   Result = Application.Run("UpdateSheet", MonthNum)
' ...[код]...
End Sub

Sub UpdateSheet(MonthSeq)
' ...[код]...
End Sub
```

В этом примере используется метод `Run`, выполняющий процедуру `UpdateSheet` и передающий `MonthNum` в качестве аргумента;

6) с помощью кнопки на панели инструментов  ;

7) из специально разработанного меню;

8) по выполнению определенного события. Такими событиями могут выступать открытие рабочей книги, сохранение рабочей книги, закрытие рабочей книги, изменение ячейки, переход на другой рабочий лист и т. д. Процедура, которая выполняется когда происходит какое-либо событие, называется *обработкой события*. Процедуры обработки событий характеризуются общими свойствами:

– имеют специальные названия, состоящие из имени объекта, символа подчеркивания и названия события. Например, процедура, которая выполняется при открытии рабочей книги, называется `Workbook_Open`,

– хранятся в окне кода для определенного объекта;

9) из окна «Immediate» (окна отладки) редактора Visual Basic. Если это окно в данный момент не отображено, нажмите Ctrl+G. Окно «Immediate» выполняет операторы VBA, которые вы вводите в его области. Чтобы выполнить процедуру, достаточно ввести название процедуры в окне «Immediate» и нажать Enter.

6.2.3. Передача аргументов в процедуры

Аргументы обеспечивают процедуру данными, использующимися в ее инструкциях. Аргумент может передавать следующие данные:

– переменная;

– константа;

– массив;

– объект.

Что касается аргументов, процедуры подобны функциям Excel в следующем:

– процедура может не принимать аргументы;

– процедура может иметь фиксированное количество аргументов;

– процедура может иметь неопределенное количество аргументов;

– не все аргументы процедуры являются обязательными;

– все аргументы могут быть необязательными.

Например, существует несколько функций Excel, не имеющих аргументов (например, `СЛЧИС`). Другие функции (такие как `СЧЕТЕСЛИ`) принимают два аргумента. Функции еще одной группы (например, `СУММ`) мо-

гут иметь неопределенное количество аргументов (до 30). Некоторые функции Excel имеют необязательные аргументы. Например, функция ПЛТ может иметь пять аргументов (три обязательных и два необязательных).

Большинство процедур, с которыми мы сталкивались ранее, объявлялись без аргументов: вводилось ключевое слово Sub, после которого указывалось название процедуры и пустые скобки. Пустые скобки обозначают, что процедура не имеет аргументов.

В примере, приведенном далее, отображается две процедуры. Процедура `Main` вызывает процедуру `ProcessFile` трижды (оператор `Call` задается в цикле `For-Next`). Однако перед вызовом `ProcessFile` создается трехэлементный массив. Внутри цикла каждый элемент массива становится аргументом вызываемой процедуры. Процедура `ProcessFile` принимает один аргумент (с названием `TheFile`). Обратите внимание, что аргумент указывается в скобках в операторе `Sub`. После выполнения `ProcessFile` программа продолжается с оператора, указанного после оператора `Call`.

```
Sub Main
   File(1) = "dept1.xls"
   File(2) = "dept2.xls"
   File(3) = "dept3.xls"
   For i = 1 To 3
         Call ProcessFile(File(i))
   Next i
End Sub
Sub ProcessFile(TheFile)
   Workbooks.Open FileName:=TheFile
' ...[код]...
End Sub
```

Существует два способа передачи аргументов в процедуру:

– по ссылке;

– по значению.

При передаче аргумента по ссылке (этот метод используется по умолчанию) в процедуру передается адрес хранения переменной в памяти. При передаче аргумента по значению в процедуру передается копия исходной переменной. Следовательно, изменение аргумента в процедуре не вызывает изменения исходной переменной.

Следующий пример подтверждает высказанную концепцию. Аргумент процедуры `Process` передается по ссылке (по умолчанию). После того как процедура `Main` присваивает переменной `MyValue` значение 10,

она вызывает процедуру Process и передает MyValue в качестве аргумента. Процедура Process умножает значение своего аргумента (с названием YourValue) на 10. По окончании процедуры Process возобновляется выполнение процедуры Main, а функция MsgBox отображает MyValue: 100.

```
Sub Main
   MyValue = 10 Call Process(MyValue)
   MsgBox MyValue
End Sub

Sub Process(YourValue)
   YourValue = YourValue * 10
End Sub
```

Если необходимо, чтобы вызываемая процедура не изменяла переменные, полученные как аргументы, то следует изменить список аргументов вызываемой процедуры так, чтобы аргументы передавались по значению, а не по ссылке. Для этого перед аргументом добавляется ключевое слово ByVal. Тогда вызываемая процедура будет управлять копией переданных данных, а не самими данными. В следующей процедуре, например, изменения, которые происходят с YourValue в процедуре Process, не влияют на значение переменной MyValue в процедуре Main. В результате функция MsgBox отображает 10, а не 100.

```
Sub Process(ByVal YourValue)
   YourValue = YourValue * 10
End Sub
```

В большинстве случаев вполне подходит метод, используемый по умолчанию, – передача аргументов по ссылке. Однако если процедура призвана изменить данные, которые передаются ей в виде аргументов, а первоначальные данные должны остаться неизмененными, то необходимо использовать передачу данных по значению.

Аргументы процедуры могут быть самыми разными и передаваться и по значению, и по ссылке. Все аргументы, перед которыми указано ключевое слово ByVal, передаются по значению; остальные – по ссылке.

Если в процедуре применена переменная с пользовательским типом данных, то ее нужно передавать по ссылке. При попытке передать ее по значению происходит ошибка.

Так как в предыдущих примерах не был объявлен тип данных ни для одного аргумента, то все аргументы имеют тип данных Variant. Но про-

цедура, использующая аргументы, может определять типы данных непосредственно в списке аргументов. Ниже представлен оператор `Sub` для процедуры с двумя аргументами, имеющими различные типы данных. Первый аргумент объявлен как `Integer`, второй – как `String`.

```
Sub Process(Iterations As Integer, TheFile As String)
```

При передаче аргументов в процедуру важно, чтобы данные, которые передаются в качестве аргументов, имели тип данных аргументов. Например, в предыдущем примере при вызове `Process` и передаче в качестве первого аргумента переменной типа `String` отображается сообщение об ошибке: *ByRef argument type mismatch*.

6.2.4. Обработка ошибок

При выполнении процедуры VBA могут происходить ошибки:

– синтаксические (которые необходимо исправить перед тем, как выполнять процедуру)

– ошибки выполнения (они происходят в процессе выполнения процедуры). В этом разделе рассматривается вторая группа ошибок.

Чтобы процедуры обработки ошибок выполнялись, следует отключить параметр «Break on All Errors». В VBE нужно выполнить команду *меню Tools → Options*, а затем щелкнуть на вкладке «General» диалогового окна «Options». Если переключатель «Break on All Errors» установлен, то VBA игнорирует процедуры обработки ошибок. Поэтому обычно используется параметр «Break on Unhandled Errors» (рис. I.6.3).

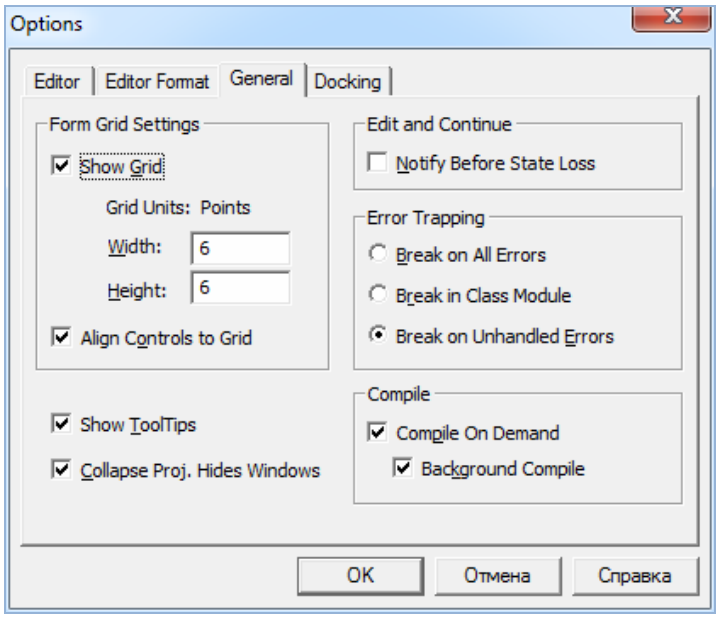


Рис. I.6.3. Вкладка «General» окна диалога «Options»

Как правило, ошибка в процессе выполнения вызывает остановку программы VBA, а пользователь видит диалоговое окно, в котором отображается номер и описание ошибки.

Перехват ошибок. Чтобы указать программе, что должно произойти при возникновении ошибки, используется оператор `On Error`. Пользователь вправе выбрать один из двух вариантов:

– проигнорировать ошибку и позволить VBA продолжить выполнение программы. После этого можно проанализировать объект `Err`, чтобы узнать, какая ошибка произошла, и при необходимости принять меры по ее предотвращению;

– перейти к специальному разделу кода для обработки ошибок, чтобы выполнить необходимые действия. Этот раздел вводится в конце процедуры и обозначается специальной меткой.

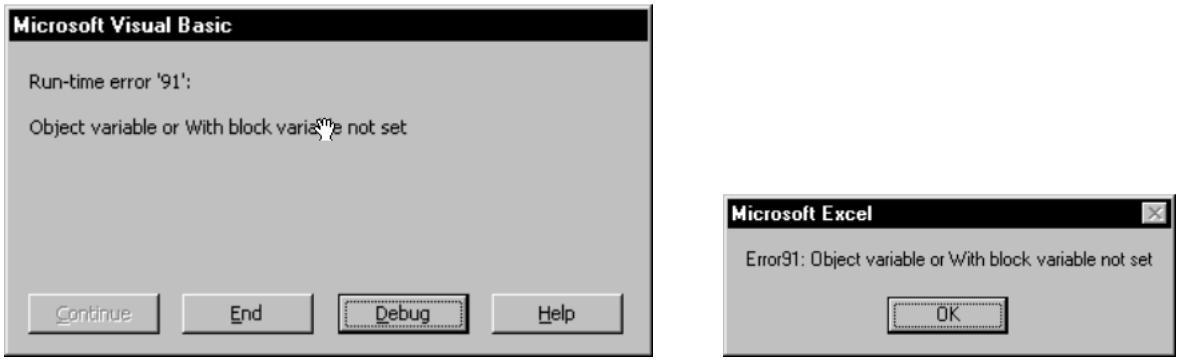
Чтобы программа продолжала выполняться после возникновения ошибки, необходимо вставить в код следующий оператор:

```
On Error Resume Next
```

Некоторые ошибки несущественны, и их можно игнорировать. Однако сначала следует определить, какая ошибка произошла. При возникновении ошибки можно использовать объект `Err` для определения ее номера. Чтобы отобразить значение `Err.Number` (по умолчанию используется в `Err`), нужно обратиться к функции `Error`. Например, представленный далее оператор отображает ту же информацию, что и обычное диалоговое окно со сведениями об ошибке Visual Basic (содержит номер ошибки и ее описание).

```
MsgBox "Ошибка" & Err & ": " & Error(Err)
```

На рисунке I.6.4, а) показано сообщение об ошибке VBA, а на рисунке I.6.4, б) отображена та же ошибка, но уже в Excel. Кроме того, можно сделать сообщение об ошибке более значимым, используя описательный текст.



а б

Рис. I.6.4. Сообщение об ошибке: а – в VBA; б – в Excel

Оператор `On Error` также применяется для определения места в процедуре, к которому должна перейти программа в случае ошибки. Чтобы обозначить это место, используется метка:

```
On Error GoTo ErrorHandler
```

Примеры обработки ошибок. В первом примере демонстрируется ошибка, которую можно проигнорировать. Метод `SpecialCells` выделяет ячейку, которая соответствует заданному критерию (действие, эквивалентное выполнению команды *меню Правка → Перейти* и щелчку на кнопке Выделить в окне диалога «Переход», чтобы выделить, например, все ячейки, содержащие формулы).

В приведенном далее примере метод `SpecialCells` выделяет все ячейки в текущем диапазоне, которые содержат формулу, возвращающую число. Если ни одна ячейка в диапазоне не соответствует критерию, VBA генерирует сообщение об ошибке. С помощью оператора `On Error Resume Next` мы предотвращаем появление сообщения об ошибке.

```
Sub SelectFormulas()
   On Error Resume Next
   Selection.SpecialCells(xlFormulas, xlNumbers).Select
   On Error GoTo 0
' ...[код]
End Sub
```

Обратите внимание, что оператор `On Error GoTo 0` восстанавливает нормальную обработку ошибок перед выходом из процедуры.

Следующая процедура использует дополнительный оператор для определения результата: произошла ли ошибка.

```
Sub SelectFormulas2()
   On Error Resume Next
   Selection.SpecialCells(xlFormulas, xlNumbers).Select
   If Err <> 0 Then MsgBox "Не было найдено ячеек с формулами."
   On Error GoTo 0
' ...[код]
End Sub
```

Если значение `Err` не равно нулю, то произошла ошибка, и в диалоговом окне отображается сообщение для пользователя.

Далее продемонстрируем обработку ошибки в результате перехода к метке.

```
Sub ErrorDemo()
   On Error GoTo Handler Selection.Value = 123
   Exit Sub
Handler:
   MsgBox "Невозможно присвоить значение выделенному диапазону."
End Sub
```

В процедуре производится попытка присвоить значение текущему выделенному объекту. Если происходит ошибка (например, не выделен диапазон ячеек или лист защищен), то оператор присвоения выдает ошибку. Оператор `On Error` задает переход к метке `Handler` в случае ошибки. Обратите внимание, что перед меткой используется оператор `Exit Sub`. Программа обработки не выполняется, если ошибок не было. Этот оператор можно не задать – в результате сообщение будет отображаться даже в том случае, если ошибка не происходила.

Иногда ошибка помогает получить определенную информацию. В приведенном ниже примере выполняется проверка того, открыта ли конкретная рабочая книга. При этом обработка ошибок не выполняется.

```
Sub CheckForFile()
   FileName = "BUDGET.XLS"
   FileExists = False

     ' Цикл по всем рабочим книгам
   For Each book In Workbooks
      If UCase(book.Name) = FileName Then
         FileExists = True
      End If
   Next book

      ' Отображение соответствующего сообщения
   If FileExists Then _
      MsgBox FileName & " открыт." Else _
         MsgBox FileName & " не открыт."
End Sub
```

В этой процедуре в цикле `For Each-Next` просматриваются все объекты коллекции `Workbooks`. Если книга открыта, переменная `FileExists` получает значение `True`. В результате отображается сообщение, указывающее пользователю, открыта ли рабочая книга.

Предыдущую процедуру можно переписать так, что для определения «открытости» будет использоваться метод обработки ошибок. В следующем примере оператор `On Error Resume Next` указывает VBA игнорировать любые ошибки. В инструкции мы обратимся к рабочей книге, присвоив ей переменную объекта (с помощью ключевого слова `Set`). Если рабочая книга закрыта, происходит ошибка. В структуре `If-Then-Else` проверяется значение свойства `Err` и отображается соответствующее сообщение.

```
Sub CheckForFile()
   Dim x as Workbook
   FileName = "BUDGET.XLS"
   On Error Resume Next
   Set x = Workbooks(FileName)
   If Err = 0 Then
      MsgBox FileNAme & " открыт."
   Else
      MsgBox FileNAme & " не открыт."
   End If
   On Error GoTo 0
End Sub
```

6.3. Работа с пользовательскими формами

6.3.1. Создание собственных диалоговых окон

Диалоговые окна являются одним из наиболее важных элементов пользовательского интерфейса в программах Windows. Практически все программы в операционной системе Windows используют такие окна. Разработчики, создающие приложения Excel, могут программировать собственные диалоговые окна с помощью объектов `UserForm`. В этом разделе будут рассмотрены следующие задачи:

– использование окна ввода данных для получения информации от пользователя;

– использование окна сообщения для отображения сведений или эмуляции ответной реакции;

– выбор файла из диалогового окна;

– выбор папки;

– отображение диалоговых окон, встроенных в Excel.

Перед тем как приступить к изучению принципов создания диалоговых окон на основе пользовательских форм `UserForm`, имеет смысл научиться использовать некоторые встроенные инструменты Excel, предназначенные для вывода диалоговых окон.

Итак, в некоторых случаях можно избежать создания собственного диалогового окна, воспользовавшись одним из нескольких встроенных окон:

– окном ввода данных;

– окном сообщения;

– окном выбора файла;

– окном ввода имени файла и его расположения при сохранении;

– окном выбора папки;

– окном запроса данных.

Использование окна ввода данных. *Окно ввода данных* – это простое диалоговое окно, которое позволяет пользователю ввести единственное значение. Например, можно применить окно ввода данных, чтобы предоставить пользователю возможность ввести текст, числа или даже диапазон значений. Для создания окна ввода используются две функции `InputBox`: одна в VBA, а вторая в Excel.

Функция InputBox в VBA имеет следующий синтаксис:

`InputBox(`*`Запрос`*`[, `*`Заголовок`*`][, `*`По_умолчанию`*`][, `*`xpos`*`][, `*`ypos`*`][, `*`Справка`*`, `*`Раздел`*`])`

– *`Запрос`* – указывает текст, отображаемый в окне ввода (обязательный параметр);

– *`Заголовок`* – определяет заголовок окна ввода (необязательный параметр);

– *`По_умолчанию`* – задает значение, которое отображается в окне ввода по умолчанию (необязательный параметр);

– *`xpos`*, *`ypos`* – определяют координаты верхнего левого угла окна ввода на экране (необязательные параметры);

– *`Справка`*, *`Раздел`* – указывают файл и раздел в справочной системе (необязательные параметры).

Функция `InputBox` запрашивает у пользователя единственное значение. Она всегда возвращает строку, поэтому результат необходимо преобразовать в числовое значение.

Текст, отображаемый в окне ввода, может достигать 1024 символов (длину допускается изменять в зависимости от ширины используемых символов). Кроме этого, можно указать заголовок диалогового окна, значение по умолчанию и координаты окна ввода на экране. Также в данном коде указывается раздел справочной системы со всеми вспомогательными сведениями. Если определить этот раздел, то в диалоговом окне будет отображена кнопка Справка.

В следующем примере, показанном на рисунке I.6.5, используется функция VBA `InputBox`, которая запрашивает у пользователя полное имя (имя и фамилию). Затем программа выделяет имя и отображает приветствие в окне сообщения.

Рис. I.6.5. Результат выполнения функции VBA InputBox

```
Sub GetName()
   Dim UserName As String
   Dim FirstSpace As Integer
   Do Until UserName <> ""
      UserName = InputBox("Введите свое имя: ", "Кто Вы?")
   Loop
   FirstSpace = InStr(UserName, " ")
   If FirstSpace <> 0 Then
      UserName = Left(UserName, FirstSpace - 1)
   End If
   MsgBox "Привет, " & UserName
End Sub
```

Обратите внимание: функция `InputBox` вызывается в цикле `Do Until`. Это позволяет убедиться в том, что данные введены в окно. Если пользователь щелкнет на кнопке Отмена (Cancel) или не введет текст, то переменная `UserName` будет содержать пустую строку, а окно ввода данных появится повторно. Далее в процедуре будет осуществлена попытка получить имя пользователя путем поиска первого символа пробела (для этого используется функция `InStr`). Таким образом, можно воспользоваться функцией `Left` для получения всех символов, расположенных слева от символа пробела. Если символ пробела не найден, то используется все введенное имя.

Как отмечалось ранее, функция `InputBox` всегда возвращает строку. Если строка, предоставленная в качестве результата выполнения функции `InputBox`, выглядит как число, ее можно преобразовать с помощью функции VBA `Val`. В противном случае следует применить функцию `InputBox` в Excel.

Использование метода Excel `InputBox` (вместо функции VBA `InputBox`) предоставляет три преимущества:

– можно задать тип возвращаемого значения;

– можно указать диапазон листа, выделив мышью ячейки листа;

– производится автоматическая проверка правильности введенных данных.

В данном случае метод `InputBox` имеет следующий синтаксис:

`object.InputBox(`*`Запрос`*`, `*`Заголовок`*`, `*`По_умолчанию`*`, `*`Слева`*`, `*`Сверху`*`, `*`Справка`*`, `*`Раздел`*`, `*`Тип`*`)`

– *`Запрос`* – указывает текст, отображаемый в окне ввода (обязательный параметр).

– *Заголовок* – определяет заголовок окна ввода (необязательный параметр).

– *По_умолчанию* – задает значение, которое отображается в окне ввода по умолчанию (необязательный параметр).

– *Слева*, *Сверху* – определяют координаты верхнего левого угла окна ввода на экране (необязательные параметры).

– *Справка*, *Раздел* – указывают файл и раздел в справочной системе (необязательные параметры).

– *Тип* – указывает код типа данных, который будет возвращаться методом (необязательный параметр). Его возможные значения перечислены в таблице I.6.6.

Таблица I.6.6.

Коды типов, возвращаемых методом Excel `InputBox` данных

Код	Значение
0	Формула
1	Число
2	Строка (текст)
4	Булево значение (истина или ложь)
8	Ссылка на ячейку как объект диапазона
16	Значение ошибки (такое как #н/д)
64	Массив значений

Метод Excel `InputBox` является достаточно гибким. Использование суммы приведенных выше значений позволяет возвратить несколько типов данных. Например, для отображения окна ввода, которое принимает текстовый или числовой тип данных, нужно установить код в значение 3 (т.е. 1+2 или «число»+«текст»). Если в качестве кода типа данных применить значение 8, то пользователь сможет ввести в поле адрес ячейки или диапазона ячеек. Кроме того, пользователь имеет возможность указать диапазон на текущем рабочем листе.

Процедура `EraseRange`, которая приведена ниже, использует метод `InputBox`. Таким образом, пользователь может указать удаляемый диапазон (рис. I.6.6). Адрес диапазона вводится вручную, мышь необходима для выделения диапазона на листе.

Метод `InputBox` с кодом 8 возвращает объект `Range` (обратите внимание на ключевое слово `Set`). После этого выбранный диапазон очищается (с помощью метода `Clear`). По умолчанию в поле окна ввода отображается адрес текущей выделенной ячейки. Если в окне ввода щелкнуть на кнопке Отмена (Cancel), то оператор `On Error` завершит процедуру.

```
Sub EraseRange()
   Dim UserRange As Range
   DefaultRange = Selection.Address
   On Error GoTo Canceled
   Set UserRange = Application.InputBox _
      (Prompt := "Удаляемый диапазон:", _
      Title := "Удаление диапазона", _
      Defaul t:= DefaultRange, _
      Type := 8)
   UserRange.Clear
   UserRange.Select
   Canceled:
End Sub
```

Еще одним преимуществом применения метода Excel `InputBox` является автоматическая проверка правильности введенных данных программой Excel. Если в примере `GetRange` ввести данные, не представляющие диапазон адресов, то Excel отобразит специальное сообщение и позволит пользователю повторить ввод данных (рис. I.6.7).

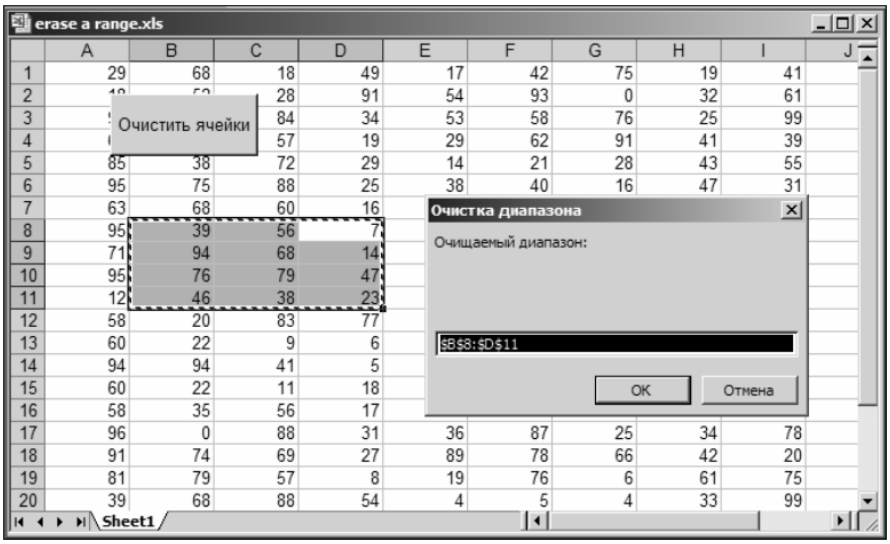


Рис. I.6.6. Использование метода InputBox для выделения и удаления диапазона

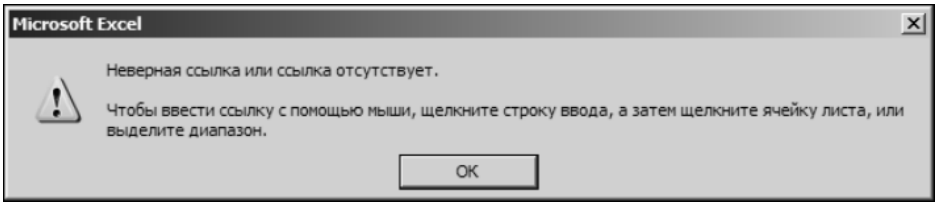


Рис. I.6.7. Метод Excel `InputBox` производит автоматическую проверку правильности введенных данных

Функция VBA MsgBox. Функция VBA `MsgBox` представляет пользователю простой способ отображения сообщения. Также эта функция задает ответную реакцию пользователя на запрос (передает результат щелчка на кнопке OK или Отмена). Ниже приведен синтаксис функции `MsgBox`:

`MsgBox(`*`Запрос`*`[, `*`Кнопки`*`][, `*`Заголовок`*`][, `*`Справка`*`, `*`Раздел`*`])`

Запрос – определяет текст, который будет отображаться в окне сообщения (обязательный параметр);

Кнопки – содержит числовое выражение, которое определяет кнопки, отображаемые в окне сообщения (необязательный параметр). Возможные значения приводятся в таблице I.6.7;

Заголовок – содержит заголовок окна сообщения (необязательный параметр);

Справка, *Раздел* – указывают файл и раздел справочной системы (необязательные параметры).

Окна сообщений можно изменять. Как правило, для этого применяется параметр *Кнопки*. С его помощью указываются отображаемые кнопки, отмечается необходимость использования значка, определяется кнопка по умолчанию. В таблице I.6.7 приведен список констант, которые можно использовать в качестве значений этого параметра.

Таблица I.6.7

Константы, используемые для выбора кнопок в функции MsgBox

Константа	Значение	Описание
`vbOKonly`	0	Отображает только кнопку OK
`vbOKCancel`	1	Отображает кнопки OK и Отмена
`vbAbortRetryIgnore`	2	Отображает кнопки Прервать, Повтор и Пропустить
`vbYesNoCancel`	3	Отображает кнопки Да, Нет и Отмена
`vbYesNo`	4	Отображает кнопки Да и Нет
`vbRetryCancel`	5	Отображает кнопки Повтор и Отмена
`vbCritical`	16	Отображает значок важного сообщения
`vbQuestion`	32	Отображает значок важного запроса
`vbExclamation`	48	Отображает значок предупреждающего сообщения
`vbInformation`	64	Отображает значок информационного сообщения
`vbDefaultButton1`	0	По умолчанию выделена первая кнопка
`vbDefaultButton2`	256	По умолчанию выделена вторая кнопка

Продолжение таблицы I.6.7

Константа	Значение	Описание
vbDefaultButton3	512	По умолчанию выделена третья кнопка
vbDefaultButton4	768	По умолчанию выделена четвертая кнопка
vbSystemModal	4098	Все приложения приостанавливают свою работу до момента, пока пользователь ответит на запрос в окне сообщения (работает не во всех случаях)

Функцию MsgBox можно использовать в качестве процедуры (для отображения сообщения), а также присвоить возвращаемое этой функцией значение переменной. Функция MsgBox возвращает результат, который представляет кнопку, на которой щелкнул пользователь. Следующий пример отображает сообщение и не возвращает результат:

```
Sub MsgBoxDemo()
   MsgBox "Щелкните на кнопке OK для продолжения"
End Sub
```

Чтобы получить результат из окна сообщения, следует присвоить возвращаемое функцией MsgBox значение переменной. В следующем коде используется ряд встроенных констант (табл. I.6.8), которые упрощают управление возвращаемыми функцией MsgBox значениями.

```
Sub GetAnswer()
   Ans = MsgBox("Продолжить?"/ vbYesNo) Select Case Ans Case
   vbYes
'     ...[код в случае Ans равно Yes]...
   Case vbNo
'     ...[код в случае Ans равно No]...
   End Select
End Sub
```

Таблица I.6.8

Константы, возвращаемые функцией MsgBox

Константа	Значение	Нажатая кнопка
vbOK	1	OK
vbCancel	2	Отмена
vbAbort	3	Прервать
vbRetry	4	Повтор
vbIgnore	5	Пропустить
vbYes	6	Да
vbNo	7	Нет

Метод Excel `GetOpenFilename`. Если приложению необходимо получить от пользователя имя файла, то можно воспользоваться функцией `InputBox`, но этот подход часто приводит к возникновению ошибок. Более надежным считается использование метода `GetOpenFilename` объекта `Application`, который позволяет удостовериться, что приложение получило корректное имя файла (а также его полный путь).

Данный метод отображает стандартное диалоговое окно «Открытие документа» (которое появляется при выполнении команды *меню Файл → → Открыть*), но при этом указанный файл не открывается. Вместо этого метод возвращает строку, которая содержит путь и имя файла, выбранных пользователем. По окончании данного процесса с именем файла можно делать все что угодно. Этот метод имеет следующий синтаксис (все параметры необязательные):

```
object.GetOpenFilename(Фильтр_файла, Индекс_фильтра, Заголовок,
Подпись_кнопки, Множественный_выбор)
```

– *Фильтр_файла* – содержит строку, определяющую критерий фильтрации файлов (необязательный параметр);

– *Индекс_фильтра* – указывает индексный номер того критерия фильтрации файлов, который используется по умолчанию (необязательный параметр);

– *Заголовок* – содержит заголовок диалогового окна (необязательный параметр). Если этот параметр не указать, то будет использован заголовок «Открытие документа»;

– *Подпись_кнопки* – применяется только в компьютерах Macintosh;

– *Множественный_выбор* – необязательный параметр. Если он имеет значение `ИСТИНА`, можно выбрать несколько имен файлов. По умолчанию данный параметр имеет значение `ЛОЖЬ`.

Аргумент *Фильтр_файла* определяет содержимое раскрывающегося списка «Тип файлов». Аргумент состоит из строки, определяющей отображаемое в диалоговом окне значение, а также строки действительной спецификации типа файлов, в которой находятся групповые символы. Оба элемента аргумента разделены запятыми. Если этот аргумент не указать, то будет использовано значение по умолчанию:

```
" Все файлы (*.*),*.*"
```

Обратите внимание на первую часть строки Все файлы (*.*). Это текст, отображаемый в раскрывающемся списке «Тип файлов». Вторая часть строки *.* указывает тип отображаемых файлов.

В следующих инструкциях переменной Filt присваивается строковое значение. Эта строка впоследствии используется в качестве аргумента Фильтр_файла метода GetOpenFilename. В данном случае диалоговое окно предоставит пользователю возможность выбрать один из четырех различных типов файлов (кроме варианта «Все файлы»). Если задать значение переменной Filt, то будет использоваться оператор конкатенации строки VBA. Этот способ упрощает управление громоздкими и сложными аргументами.

```
Filt = "Текстовые файлы (*.txt),*.txt," & _
   "Файлы печати(*.prn),*.prn," & _
   "Разделенные запятой(*.csv),*.csv," & _
   "ASCII (*.asc),*.asc," & _
"Все файлы(*.*),*.*"
```

Аргумент Индекс_фильтра указывает значение аргумента Фильтр_файла по умолчанию. Аргумент Заголовок определяет текст, который отображается в заголовке окна. Если параметр Множественный_выбор имеет значение ИСТИНА, то пользователь может выбрать в окне несколько файлов. Имя каждого файла заносится в массив.

В следующем примере у пользователя запрашивается имя файла. При этом в поле типа файлов используется пять фильтров.

```
Sub GetImportFileName()
   Dim Filt As String
   Dim FilterIndex As Integer
   Dim FileName As Variant
   Dim Title As String
' Настройка списка фильтров
Filt = "Текстовые файлы (*.txt),*.txt," & _
   "Файлы печати(*.prn),*.prn," & _
   "Разделенные запятой(*.csv),*.csv," & _
   "ASCII (*.asc),*.asc," & _
"Все файлы(*.*),*.*"
' По умолчанию используется фильтр *.*
  FilterIndex = 5
' Заголовок окна
  Title = "Выберите импортируемый файл"
```

```
' Получение имени файла
  FileName = Application.GetOpenFilename _
     (FileFilter:=Filt, _
     FilterIndex:=FilterIndex, _
     Title:=Title)
' При отмене выйти из окна
   If FileName = False Then
     MsgBox "Файл не выбран"
     Exit Sub
   End If
' Отображение полного имени и пути
   MsgBox "Вы выбрали " & FileName
End Sub
```

На рисунке I.6.8 показано диалоговое окно, которое выводится на экран при выполнении этой процедуры.

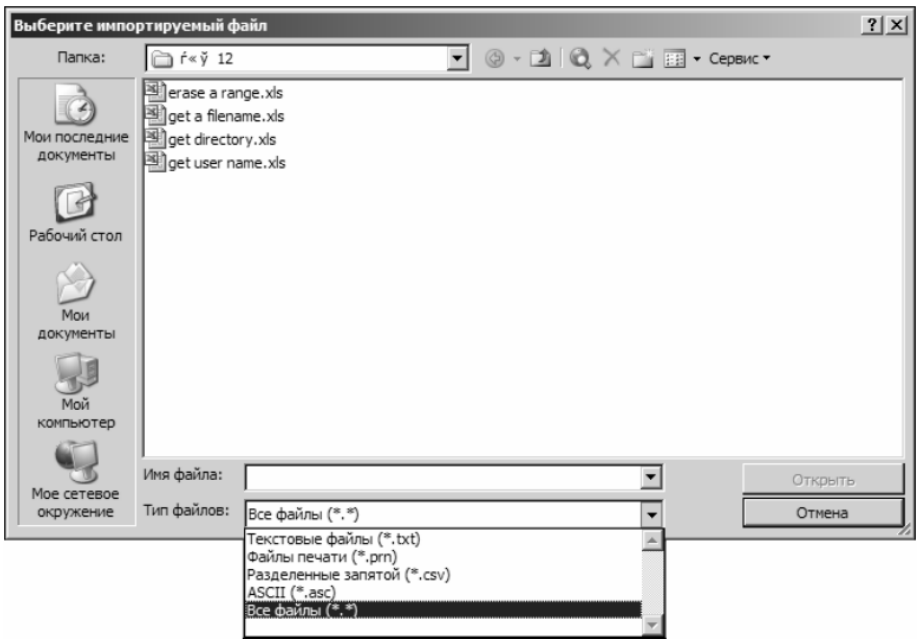


Рис. I.6.8. Метод `GetOpenFilename` отображает пользовательское диалоговое окно

Метод Excel `GetSaveAsFilename`. Данный метод имеет много общего с методом `GetOpenFilename`. Он отображает диалоговое окно «Сохранение документа» и позволяет пользователю выбрать (или указать) имя сохраняемого файла. В результате возвращается имя файла, но никакие действия не предпринимаются.

Этот метод имеет следующий синтаксис:

`object.GetSaveAsFilename (`*`Начальное_имя`*`, `*`Фильтр_файла`*`, `*`Индекс_фильтра`*`, `*`Заголовок`*`, `*`Текст_кнопки`*`)`

Начальное_имя – указывает предполагаемое имя файла (необязательный параметр);

Фильтр_файла – содержит критерий фильтрации отображаемых в окне файлов (необязательный параметр);

Индекс_фильтра – код критерия фильтрации файлов, который используется по умолчанию (необязательный параметр);

Заголовок – определяет текст заголовка диалогового окна (необязательный параметр);

Текст_кнопки – предназначен только для платформ Macintosh.

Получение имени папки. Для выбора папки в Excel 2002 и выше используется объект `FileDialog`.

Представленная ниже процедура отображает диалоговое окно, которое позволяет пользователю выбрать папку. Выбранная папка (или сообщение «Отменено») отображается с помощью функции `MsgBox`.

```
Sub GetAFolder2()
      With Application.FileDialog(msoFileDialogFolderPicker)
      .InitialFileName = Application.DefaultFilePath & "\"
      .Title = "Укажите расположение резервной копии"
      .Show
      If .SelectedItems.Count = 0 Then
         MsgBox "Отменено"
      Else
         MsgBox .SelectedItems(1)
      End If
   End With
End Sub
```

Объект `FileDialog` позволяет указать начальную папку. Для этого необходимо задать значение свойства `InitialFileName`. В нашем случае в качестве начальной папки по умолчанию используется папка сохранения файлов Excel.

Отображение встроенных диалоговых окон Excel. VBA может выполнять команды меню Excel. Если в процессе выполнения кода VBA на экране отображаются диалоговые окна, то в коде можно автоматически «выбирать опции» диалогового окна (хотя при этом само диалоговое окно и не отображается). Например, следующий оператор VBA аналогичен выполнению команды *меню Правка → Перейти*, выделению диапазона `A1:C3` и щелчку на кнопке OK. Однако в данном случае диалоговое окно «Переход» не отображается (это именно то, что необходимо).

```
Application.Goto Reference:=Range("A1:C3")
```

В некоторых случаях может понадобится отобразить одно из встроенных диалоговых окон Excel, чтобы пользователь мог выбрать в нем необходимые параметры. Для этого можно воспользоваться одним из двух способов:

– получить доступ к коллекции `Dialogs` объекта `Application`;

– непосредственно выбрать соответствующую команду меню.

Использование коллекции `Dialogs`. Коллекция `Dialogs` объекта `Application` состоит из 258 элементов, которые представляют большую часть встроенных диалоговых окон Excel. Каждому элементу соответствует предопределенная константа, которая позволяет указать, какое именно диалоговое окно необходимо отобразить. Например, диалоговое окно Excel «Переход» представлено константой `xlDialogFormulaGoto`.

Воспользуемся методом `Show` для фактического отображения диалогового окна. Ниже приведен пример, в котором отображается диалоговое окно «Переход» (рис. I.6.9).

```
Application.Dialogs(xlDialogFormulaGoto).Show
```

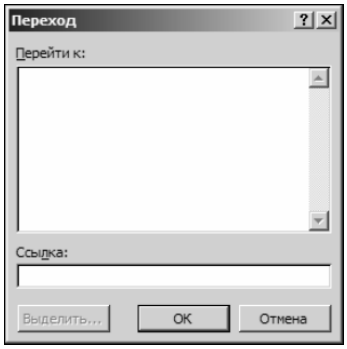


Рис. I.6.9. Диалоговое окно, выведеное оператором VBA

Когда на экране отображается диалоговое окно «Переход», пользователь может указать именованный диапазон или адрес ячейки, к которой необходимо перейти. Указанное диалоговое окно появляется при выполнении команды *меню Правка → Перейти* (также можно нажать клавишу F5).

Кроме того, можно написать код, который будет определять, как пользователь завершил работу в диалоговом окне. Для этого воспользуемся переменной. В представленном ниже операторе переменная `Result` будет иметь значение `True`, если пользователь щелкнет на кнопке OK, и `False`, если он щелкнет на кнопке Отмена или нажмет клавишу Esc.

```
Result = Application.Dialogs(xlDialogFormulaGoto).Show
```

Получение дополнительной информации о встроенных диалоговых окнах. Список всех констант диалоговых окон можно получить в справочной системе или с помощью средства «Object Browser». Для отображения элементов коллекции `Dialogs` в окне «Object Browser» необходимо:

10) активизировав модуль VBA, нажать клавишу F2 – отобразится окно «Object Browser»;

11) в диалоговом окне «Object Browser» выбрать в верхней части списка значение `Excel`;

12) ввести `xlDialog` во втором списке;

13) щелкнуть на кнопке с изображением бинокля.

Использование аргументов во встроенных диалоговых окнах. Большая часть встроенных диалоговых окон поддерживает передачу аргументов, которые обычно соответствуют элементам управления диалогового окна. Например, диалоговое окно «Формат ячеек» (оно вызывается с помощью константы `xlDialogCellProtection`) принимает два аргумента: `locked` и `hidden`. Если необходимо отобразить диалоговое окно с уже установленными флажками, следует воспользоваться следующим оператором:

```
Application.Dialogs(xlDialogCellProtection).Show True, True
```

Аргументы для каждого из встроенных диалоговых окон перечисляются в справочной системе к программе VBE. Для того чтобы найти необходимый раздел в справочном руководстве, нужно ввести в качестве критерия поиска фразу *Built-In Dialog Box Argument List*.

Непосредственный выбор команды меню. Команду меню можно выбрать программным образом. Это позволяет отображать диалоговое окно с помощью команды меню.

Следующий оператор аналогичен выполнению команды *меню Правка → Перейти*.

```
Application.CommandBars("Worksheet Menu Bar"). _
   Controls("Правка").Controls("Перейти...").Execute
```

Если выполнить этот оператор, будет отображено диалоговое окно «Переход». При этом все названия опций должны полностью совпадать (включая троеточие после `Перейти`).

В отличие от применения коллекции `Dialogs`, эта методика не позволяет активизировать элементы управления диалогового окна по умолчанию.

6.3.2. Работа с пользовательскими формами

Обработка пользовательских диалоговых окон в Excel. Пользовательские диалоговые окна создаются на основе технологии пользовательских форм `UserForm`, к которым можно получить доступ из редактора Visual Basic.

Ниже приведена стандартная последовательность действий, которой следует придерживаться при создании пользовательского диалогового окна:

1) вставить новую форму `UserForm` в проект `VBAProject` рабочей книги;

2) создать процедуру, которая будет отображать форму `UserForm`. Эта процедура располагается в модуле кода VBA (а не в модуле кода формы `UserForm`);

3) добавить элементы управления в форму `UserForm`;

4) настроить свойства добавленных элементов управления;

5) создать процедуры обработки событий для элементов управления. Эти процедуры добавляются в модуль кода `UserForm` и выполняются при возникновении различных событий (например, при щелчке на кнопке).

Вставка новой формы `UserForm`. Для того чтобы добавить в проект форму `UserForm`, нужно:

1) запустить VBE (Alt+F11);

2) указать рабочую книгу в окне проекта;

3) выполнить команду *меню Insert → UserForm*.

Формы `UserForm` получат такие имена, как `UserForm1`, `UserForm2` и т.д.

Чтобы упростить идентификацию, можно изменить имя формы `UserForm`:

1) активизировать форму;

2) воспользоваться окном «Properties» для изменения свойства `Name` (нажать клавишу F4, если окно «Properties» не отображается на экране).

На рисунке I.6.10 приведен пример окна «Properties» для новой пустой формы `UserForm`.

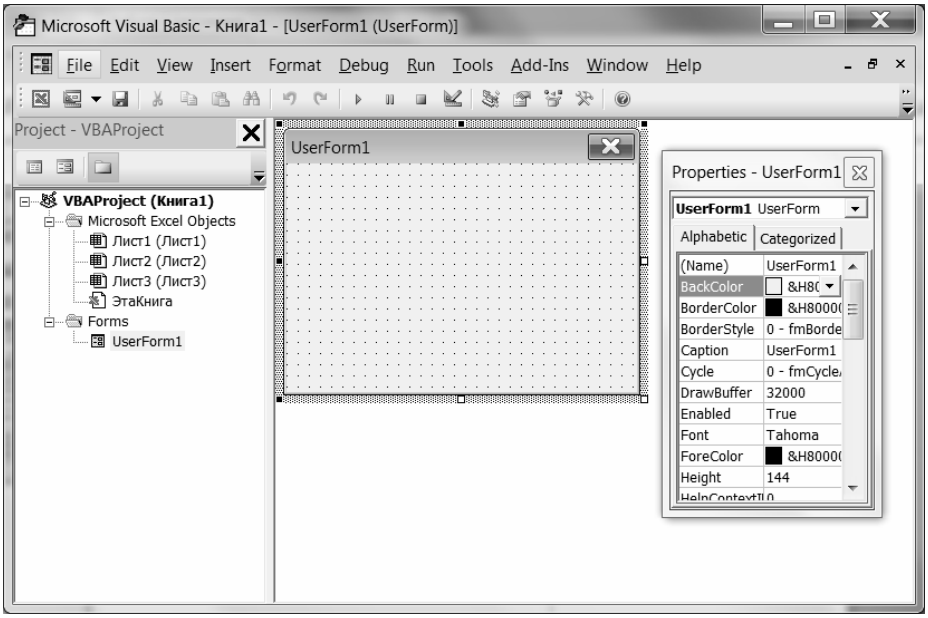


Рис. I.6.10. Окно Properties для пустого пользовательского диалогового окна

Рабочая книга может содержать любое количество пользовательских диалоговых окон. При этом каждая форма `UserForm` соответствует лишь одному пользовательскому диалоговому окну.

Добавление элементов управления в пользовательское диалоговое окно. Чтобы добавить элементы управления в форму `UserForm`, следует воспользоваться окном «Toolbox»:

1) если окно «Toolbox» не отображено на экране, выполнить команду *меню View → Toolbox*. Окно «Toolbox» показано на рисунке I.6.11;

2) щелкнуть на той кнопке в окне «Toolbox», которая соответствует добавляемому элементу управления;

3) щелкнуть внутри диалогового окна для создания элемента управления (используется размер элемента по умолчанию). Также можно щелкнуть на элементе управления и, перетащив его границы в диалоговом окне, задать необходимый размер в пользовательском диалоговом окне.

После добавления нового элемента управления ему назначается имя, которое состоит из названия типа элемента управления и числового кода. Например, если добавить элемент управления `CommandButton` в пустую форму `UserForm`, то этот элемент управления будет называться `CommandButton1`. Если добавить в окно второй элемент управления `CommandButton`, то он будет называться `CommandButton2`.

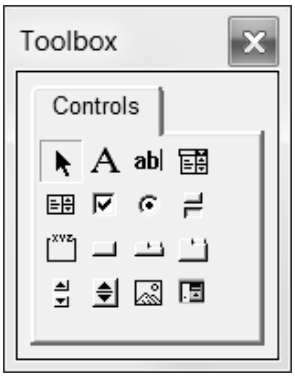


Рис. I.6.11. Окно «Toolbox»

Рекомендуется переименовывать все элементы управления, которые управляются с помощью VBA-кода. Это позволит использовать более содержательные имена объектов (например, `ProductListBox`), а не общие последовательно нумерованные названия (подобные `ListBox1`). Для того чтобы изменить имя элемента управления, используется окно «Properties» в VBE, достаточно выделить необходимый объект и ввести новое имя.

Доступные элементы управления

`CheckBox` 

Элемент управления `CheckBox` предоставляет пользователю возможность выбрать один из двух вариантов: «Да» или «Нет», «Истина» или «Ложь», «Включить» или «Выключить» и т. д. Если элемент управления `CheckBox` установлен, то он имеет значение `True`, в противном случае значение равно `False`.

ComboBox 

Элемент управления ComboBox подобен объекту ListBox. Отличие заключается в том, что ComboBox представляет раскрывающийся список, в котором в определенный момент времени отображается только одно значение. Кроме того, пользователю в поле списка разрешено вводить значение, которое не обязательно представляет одну из опций объекта ComboBox.

CommandButton 

Каждое создаваемое диалоговое окно будет иметь как минимум один элемент управления CommandButton. Обычно используются объекты CommandButton, представляющие кнопки OK и Отмена.

Frame 

Элемент управления Frame применяется в качестве оболочки для других элементов управления. Он добавляется в диалоговое окно либо в целях эстетики, либо из соображений логического группирования однотипных элементов управления. Элемент управления Frame используется в случае, когда диалоговое окно содержит более одного набора элементов управления OptionButton.

Image 

Элемент управления Image используется для представления графического изображения, которое сохранено в отдельном файле или вставляется из буфера обмена. Графическое изображение сохраняется вместе с рабочей книгой. Таким образом, при передаче рабочей книги другому пользователю передавать вместе с ней копию графического файла не обязательно.

Label 

Элемент управления Label отображает текст в диалоговом окне.

ListBox 

Элемент управления ListBox предоставляет список опций, из которого пользователь может выбрать один вариант (или несколько).

MultiPage 

Элемент управления MultiPage позволяет создавать диалоговые окна с несколькими вкладками. По умолчанию элемент управления MultiPage состоит из двух вкладок. Чтобы создать дополнительные вкладки, нужно щелкнуть правой кнопкой мыши на существующей вкладке и выбрать команду *New Page* из появившегося на экране контекстного меню.

OptionButton 

Элемент управления OptionButton применяется при выборе пользователем одного варианта из нескольких. Эти элементы управления всегда группируются в диалоговом окне в наборы, содержащие не менее двух опций. Когда один элемент управления OptionButton выбран, все остальные элементы управления OptionButton текущей группы автоматически становятся неактивными.

Если пользовательское диалоговое окно содержит более одного набора элементов управления OptionButton, то каждый из таких наборов должен иметь собственное значение свойства GroupName. В противном случае все элементы управления OptionButton в диалоговом окне рассматриваются как члены одной группы. Также можно вставить элементы управления OptionButton в объект Frame, что приведет к их автоматическому группированию в текущем разделе.

RefEdit 

Элемент управления RefEdit используется в том случае, когда пользователь должен выделить диапазон ячеек на листе.

ScrollBar 

Элемент управления ScrollBar в некотором смысле подобен элементу управления SpinButton. Разница заключается в том, что пользователь может перетаскивать бегунок объекта ScrollBar для изменения значения с большим приращением. Элемент управления ScrollBar рекомендуется использовать при выборе значения из большого диапазона.

SpinButton 

Элемент управления SpinButton позволяет выбрать значение в результате щелчка на одной из двух кнопок со стрелками. Одна из них применяется для увеличения значения, а вторая – для уменьшения. Элемент управления SpinButton часто используется совместно с элементами управления TextBox или Label, которые содержат текущее значение элемента управления SpinButton.

TabStrip 

Элемент управления TabStrip подобен элементу управления MultiPage, однако использовать его сложнее. Элемент управления

TabStrip, в отличие от MultiPage, не выступает контейнером для других объектов. Как правило, элемент управления MultiPage обладает более широкими возможностями.

TextBox 

Элемент управления TextBox позволяет пользователям вводить в диалоговом окне текст.

ToggleButton 

Элемент управления ToggleButton имеет два состояния: включен или выключен. Щелчок на кнопке приводит к изменению состояния на противоположное и к изменению внешнего вида кнопки. Этот элемент управления может иметь значение True (активен) или False (неактивен).

Настройка элементов управления пользовательского диалогового окна. После того как элемент управления будет помещен в диалоговое окно, его можно переместить, а также изменить его размер. Для этого используются стандартные методики управления графическими объектами мышью.

Существует возможность выделить несколько элементов управления. При этом должна удерживаться нажатой клавиша Shift и выполняться щелчки на объектах. Можно также обвести указателем мыши все необходимые элементы управления.

Форма UserForm содержит вертикальные и горизонтальные направляющие, которые помогают выровнять добавленные в диалоговое окно элементы управления. При добавлении или перемещении элемент управления привязывается к направляющим, что облегчает упорядочивание таких элементов в окне. Если направляющие не используются, то можно отключить их, выполнив команду *меню Tools → Options*. В диалоговом окне «Options» перейдите на вкладку «General» и выберите соответствующие опции в разделе «Form Grid Settings».

Меню «Format» окна VBE предоставляет несколько команд, которые позволяют точно разместить и выровнять элементы управления в диалоговом окне. Перед использованием этих команд необходимо указать элементы управления, к которым они будут применяться. Эти команды выполняют свои задачи так, как и ожидается. На рисунке I.6.12 показано диалоговое окно с несколькими элементами управления OptionButton в процессе выравнивания.

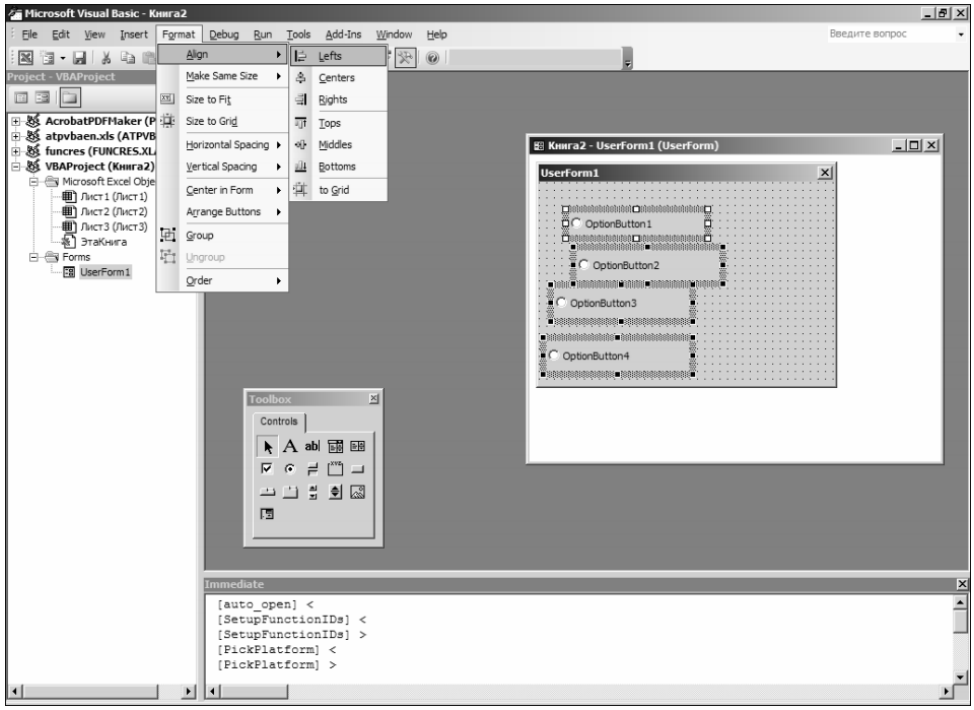


Рис. I.6.12. Диалоговое окно с несколькими элементами управления `OptionButton` в процессе выравнивания

Изменение свойств элементов управления. Каждый элемент управления характеризуется набором параметров, которые определяют внешний вид и поведение элемента управления. Свойства элемента управления можно изменять в следующих случаях:

– в момент проектирования при разработке пользовательского диалогового окна. Для этого используется окно «Properties»;

– в процессе выполнения, когда пользовательское диалоговое окно отображается на экране. Для этого используются инструкции VBA.

В VBE окно «Properties» позволяет изменять свойства выделенного элемента управления (это может быть обычный элемент управления или сама форма `UserForm`).

В окне «Properties» расположены две вкладки. Вкладка «Alphabetic» представляет свойства выделенного элемента управления в алфавитном порядке. Вкладка «Categorized» отображает свойства, сгруппированные в логические категории. Обе вкладки содержат одинаковые свойства, но в различном порядке.

Для того чтобы изменить свойство, необходимо щелкнуть на нем и ввести новое значение. Некоторые свойства могут принимать только ограниченный набор допустимых значений, выбираемых из соответствующего списка. При щелчке на таком свойстве в окне «Properties» будет отображена кнопка со стрелкой, указывающей вниз. Щелкните на этой кнопке, что-

бы выбрать значение из предложенного списка. Например, свойство `TextAlign` может принимать одно из следующих значений: `1 – fmTextAlignLeft, 2 - fmTextAlignCenter` и `3 - fmTextAlignRight`.

При выделении отдельных свойств (например, `Font` и `Picture`) рядом с ними отображается небольшая кнопка с троеточием. Щелчок на этой кнопке приводит к вызову диалогового окна настройки свойства.

Изменение порядка просмотра (активизации). Порядок просмотра определяет последовательность, в которой активизируются элементы управления при нажатии пользователем комбинаций клавиш <Tab> и <Shift+Tab>. Кроме того, порядок активизации указывает, какой элемент управления по умолчанию выделяется на форме первым. Если пользователь вводит текст в элемент управления `TextBox`, то этот элемент управления считается активным. Если далее щелкнуть на элементе управления `OptionButton`, то именно он станет активным. Элемент управления, назначенный первым для просмотра, будет активным в момент открытия диалогового окна.

Для того чтобы указать порядок активизации, следует выполнить команду *меню View → Tab Order*. Кроме того, можно щелкнуть правой кнопкой мыши на диалоговом окне и выбрать *Tab Order* из появившегося контекстного меню. В любом случае, Excel отобразит диалоговое окно «Tab Order», которое показано на рисунке I.6.13.

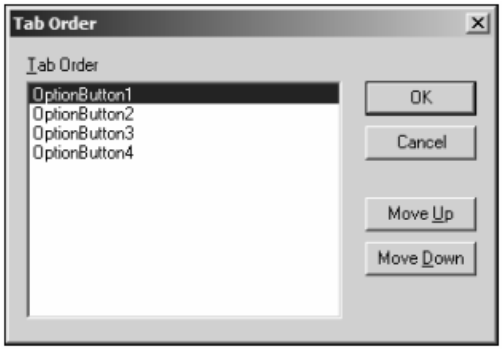


Рис. I.6.13. Окно диалога «Tab Order»

Диалоговое окно «Tab Order» содержит упорядоченный список всех элементов управления в последовательности, которая соответствует порядку активизации объектов в пользовательском диалоговом окне. Чтобы переместить элемент управления в списке, нужно выбрать его и щелкнуть на кнопке Move Up или Move Down. Можно одновременно перемещать более одного элемента управления (удерживая при выделении элементов клавишу Shift или Ctrl).

Кроме того, можно указать порядок активизации элемента управления с помощью окна «Properties». Первый активизируемый элемент управления будет иметь свойство `TabIndex`, установленное в значение `0`. Изменение значения свойства `TabIndex` текущего объекта приведет к изменению значений свойств `TabIndex` других элементов управления. Изменения вносятся автоматически. Можно удостовериться, что ни один элемент управления не имеет значение свойства `TabIndex` большее, чем количество элементов управления в диалоговом окне. Если необходимо удалить элемент управления из списка активизируемых объектов, то следует назначить его свойству `TabStop` значение `False`.

Некоторые элементы управления, такие как `Frame` и `MultiPage`, являются контейнерами для других элементов управления. Элементы управления в контейнере имеют собственный порядок активизации. Для того чтобы указать порядок активизации для элементов управления `OptionButton` внутри элемента управления `Frame`, нужно выделить последний перед выполнением команды *меню View → Tab Order*.

Назначение сочетаний клавиш. Большинству элементов управления диалогового окна можно назначить сочетание клавиш. Таким образом, пользователь получит доступ к элементу управления по нажатию Alt и указанной клавиши. Использование свойства `Accelerator` в окне «Properties» позволяет определить клавишу для активизации элемента управления.

Некоторые элементы управления не имеют свойства `Accelerator`, так как они не отображают значение свойства `Caption`. Но доступ посредством клавиатуры к таким элементам управления можно предоставить с помощью элемента управления `Label`. Назначьте клавишу элементу управления `Label`, после чего расположите его в последовательности активизации перед необходимым элементом управления `TextBox`.

Тестирование пользовательского диалогового окна. Обычно при разработке возникает необходимость в тестировании диалогового окна `UserForm`. Существует три способа, которые позволяют проверить диалоговое окно без вызова его из процедуры VBA:

– выполнить команду *меню Run → Run Sub/UserForm*;

– нажать клавишу F5.

– щелкнуть на кнопке «Run Sub/UserForm» на панели инструментов «Standard2».

Эти методы позволяют запустить событие инициализации диалогового окна. Как только диалоговое окно будет отображено в тестовом режиме, можно начинать проверку порядка активизации объектов и поддержки сочетаний клавиш.

Отображение пользовательского диалогового окна. Для того чтобы отобразить пользовательское диалоговое окно с помощью VBA, необходимо создать процедуру, которая вызывает метод `Show` объекта `UserForm`. Если объект `UserForm` называется `UserForm1`, то следующая процедура отобразит это пользовательское диалоговое окно:

```
Sub ShowDialog()
   UserForm1.Show
End Sub
```

Данная процедура должна располагаться в стандартном модуле VBA, а не в модуле формы `UserForm.`

При отображении пользовательской формы она остается на экране до тех пор, пока ее не скроют. Обычно в пользовательскую форму добавляют элемент управления `CommandButton`, который запускает процедуру закрытия формы. Эта процедура либо выгружает пользовательскую форму с помощью метода `Unload`, либо скрывает пользовательскую форму с экрана с помощью метода `Hide` объекта `UserForm`.

Закрытие пользовательского диалогового окна. Для того чтобы закрыть форму `UserForm1`, следует воспользоваться командой `Unload`.

```
Unload UserForm1
```

Также можно применить следующий оператор.

```
Unload Me
```

В этом случае ключевое слово `Me` применяется для идентификации пользовательской формы.

Объект `UserForm` может использовать метод `Hide`. При его вызове диалоговое окно исчезает, но остается в памяти, поэтому в коде можно получить доступ к различным свойствам элементов управления. Ниже приведен пример оператора, который скрывает диалоговое окно.

```
UserForm1.Hide
```

Также можно воспользоваться следующим оператором.

```
Me.Hide
```

О процедурах обработки событий. Как только диалоговое окно появляется на экране, пользователь начинает с ним взаимодействовать – выбирать опции в элементе управления `ListBox`, щелкать на кнопках `CommandButton` и т. д. Используя официальную терминологию, можно сказать, что пользователь генерирует события. Например, щелчок на элементе управления `CommandButton` приводит к возникновению события

`Click` объекта `CommandButton`. Таким образом, пользователю необходимо создать процедуры, которые будут выполняться при возникновении соответствующих событий.

Процедуры обработки событий вводятся в модуле кода объекта `UserForm`. Наряду с этим процедура обработки события может вызывать другие процедуры, которые находятся в стандартном модуле VBA.

Практическая сторона создания пользовательских форм будет рассмотрена в лабораторном практикуме.

Словарь терминов

Visual Basic for Application – это объектно-ориентированная среда, содержащая большой набор объектов, каждый из которых обладает множеством свойств и методов.

Аргумент – это информация, используемая процедурой в процессе выполнения.

Ассемблеры – языки программирования, близкие к программированию непосредственно в машинных кодах.

Коллекция – это совокупность одинаковых объектов.

Комментарий – это описательный текст, который включается в код.

Константа – это именованное значение или строка, которая никогда не меняется.

Локальная переменная – это переменная, объявленная в процедуре.

Массив – это группа элементов одного типа, которые имеют общее имя.

Метод – это действие, которое может быть выполнено над объектом.

Модуль – это объект, содержащий созданную пользователем процедуру.

Объект в Excel VBA – это любой элемент приложения и само приложение Microsoft Excel.

Окно ввода данных – это простое диалоговое окно, которое позволяет пользователю ввести единственное значение.

Оператор присвоения – это инструкция VBA, выполняющая математическое вычисление и присваивающая результат переменной или объекту.

Переменная – это именованное место хранения данных в памяти компьютера.

Переменная объекта – это переменная, представляющая целый объект, например, диапазон или рабочий лист.

Пользовательский интерфейс – это метод взаимодействия пользователя с приложением.

Программа на языке программирования – совокупность операторов, записанных в соответствии с принятым синтаксисом.

Программирование – это процесс создания последовательности действий (операций), проводимый в целях достижения требуемого результата.

Процедура – это совокупность операторов языка VBA, реализующая ряд логических шагов для выполнения конкретного действия.

Свойство – это атрибут объекта, описывающий, как объект выглядит и как он действует.

Событие – совершается всякий раз, когда пользователь взаимодействует с определенным объектом в рабочем листе.

Цикл – это процесс повторения набора инструкций.

Элементы управления – это элементы, из которых состоит диалоговое окно.

Язык программирования – это набор символов и терминов, который в соответствии с правилами синтаксиса описывает алгоритм решения задачи.

Вопросы и задания для самоконтроля

1. Что такое язык программирования?
2. Что такое программа?
3. Что представляет собой процесс программирования?
4. Из каких стадий состоит процесс программирования?
5. Охарактеризуйте различные уровни языков программирования.
6. Охарактеризуйте группы языков программирования.
7. Каковы принципы разработки приложений электронных таблиц?
8. Какие вопросы необходимо продумать на начальном этапе планирования проекта?
9. Что такое пользовательский интерфейс?
10. Какими средствами, необходимыми при разработке пользовательского интерфейса, располагает Microsoft Excel?
11. Для чего используются пользовательские диалоговые окна?
12. Как в Microsoft Excel можно настроить меню?
13. Как в Microsoft Excel можно настроить панель инструментов?
14. Как ведется работа с конечным пользователем?
15. Что такое модуль?
16. Что такое процедура?
17. Что такое Visual Basic for Application?
18. Что такое объект в Excel VBA?

19. Что такое коллекция в Excel VBA?

20. Что такое свойство?

21. Что такое метод?

22. Что такое событие

23. Какими способами осуществляется запуск среды VBE?

24. Из каких структурных элементов состоит окно редактора Visual Basic?

25. Как добавить новый модуль VBA?

26. Какие способы ввода кода VBA существуют?

27. Как выполняется настройка среды VBE?

28. Что такое комментарий?

29. Что такое переменная?

30. Каковы правила именования переменных в VBA?

31. Какие типы данных существуют в VBA?

32. Как выполняется объявление переменных?

33. Что такое локальная переменная?

34. Что такое константа?

35. Что такое оператор присвоения?

36. Какие математические операторы используются в VBA?

37. Что такое массив?

38. Что такое переменная объекта?

39. Какие конструкции упрощают управление объектами и коллекциями?

40. Каково назначение оператора `GoTo`?

41. Каково назначение конструкции `If-Then`?

42. Каково назначение конструкции `Select Case`?

43. Каково назначение конструкции `For-Next`?

44. Каково назначение конструкции `Do While`?

45. Каково назначение конструкции `Do Until`?

46. Что такое цикл?

47. Что такое аргумент?

48. Что такое окно ввода данных?

49. Для чего используется функция `MsgBox`?

50. Где выполняется настройка свойств элементов управления?

51. Как выполняется тестирование пользовательского диалогового окна?

II. СЕТЕВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

1. Компьютерные сети

1.1. Назначение компьютерных сетей

При физическом соединении двух или более компьютеров каналом связи образуется компьютерная *сеть*. В общем случае, для создания компьютерных сетей необходимо специальное аппаратное обеспечение (*сетевое оборудование*) и специальное программное обеспечение (*сетевые программные средства*). Простейшее соединение двух компьютеров для обмена данными называется *прямым соединением*.

Назначение всех компьютерных сетей состоит в обеспечении совместного доступа к общим *ресурсам*. В зависимости от назначения сети в термин *ресурс* можно вкладывать различный смысл. Ресурсы бывают трех типов: *аппаратные*, *программные* и *информационные*. Например, устройство печати (принтер), емкости жестких дисков – это аппаратный ресурс. Когда все участники небольшой компьютерной сети пользуются одним общим принтером, это значит, что они разделяют общий аппаратный ресурс. То же можно сказать и о сети, имеющей один компьютер с увеличенной емкостью жесткого диска (*файловый сервер*), на котором все участники сети хранят свои архивы и результаты работы.

Кроме аппаратных ресурсов компьютерные сети позволяют совместно использовать *программные ресурсы*. Так, например, для выполнения очень сложных и продолжительных расчетов можно подключиться к удаленной большой ЭВМ и отправить вычислительное задание на нее, а по окончании расчетов точно так же получить результат обратно.

Данные, хранящиеся на удаленных компьютерах, образуют *информационный ресурс*. Роль этого ресурса видна наиболее ярко на примере Интернета, который воспринимается, прежде всего, как гигантская информационно-справочная система.

При работе в компьютерной сети любого типа одновременно происходит совместное использование всех типов ресурсов. Так, например, обращаясь в Интернет за какой-либо информацией, мы используем аппаратные средства, на которых работают программы, обеспечивающие поставку затребованных нами данных.

1.2. Классификация компьютерных сетей

Компьютерные сети могут быть классифицированы по различным признакам, в частности по *территориальной распространенности*, по *скорости передачи данных*, по *топологии*, по *способу управления*, по *методу коммутации*.

1.2.1. Классификация по территориальной распространенности

По территориальной распространенности компьютерные сети принято разделять на *локальные* (*LAN – Local Area Network*), *глобальные* (*WAN – Wide Area Network*), *региональные* (*MAN – Metropolitan Area Network*).

Локальные компьютерные сети охватывают ограниченную территорию (обычно в пределах удаленности станций не более чем на несколько десятков или сотен метров друг от друга, реже на 1–2 км). Они могут объединять компьютеры одного помещения, этажа, здания, группы компактно расположенных сооружений. Отличительной чертой LAN является большая скорость передачи данных, низкий уровень ошибок и использование дешевой среды передачи данных. Большинство LAN принадлежат какой-либо конкретной организации, которая их поддерживает.

У участников рабочих групп могут быть разные права для доступа к общим ресурсам сети. Совокупность приемов разделения и ограничения прав участников компьютерной сети называется *политикой сети*. Управление сетевыми политиками (их может быть несколько в одной сети) называется *администрированием сети*. Лицо, управляющее организацией работы участников локальной компьютерной сети, называется *системным администратором*.

Создание локальных сетей характерно для отдельных предприятий или отдельных подразделений предприятий. Если предприятие (или отрасль) занимает обширную территорию, то отдельные локальные сети могут объединяться в глобальные сети. В этом случае локальные сети связывают между собой с помощью любых традиционных каналов связи (кабельных, спутниковых, радиорелейных и т.п.).

Для связи между собой нескольких локальных сетей, работающих по разным протоколам, служат специальные средства, называемые *шлюзами*. Шлюзы могут быть как аппаратными, так и программными. Например, это может быть специальный компьютер (*шлюзовой сервер*) или компьютерная программа (*шлюзовое приложение*). В последнем случае компьютер может выполнять не только функцию шлюза, но и какие-то иные функции, типичные для рабочих станций.

При подключении локальной сети предприятия к глобальной сети важную роль играет понятие *сетевой безопасности*. В частности, должен быть ограничен доступ в локальную сеть для посторонних лиц извне, а также ограничен выход за пределы локальной сети для сотрудников предприятия, не имеющих соответствующих прав. Для обеспечения сетевой безопасности между локальной и глобальной сетью устанавливают так называемые *брандмауэры*. Брандмауэром может быть специальный компьютер или компьютерная программа, препятствующая несанкционированному перемещению данных между сетями.

Глобальные компьютерные сети имеют, как правило, увеличенные географические размеры. Они могут объединять как отдельные компьютеры, так и отдельные локальные сети (рис. II.1.1). По сравнению с LAN большинство WAN отличают медленная скорость передачи данных и более высокий уровень ошибок.

Сравнительно недавно появились *городские сети* или *сети мегаполисов (MAN, Metropolitan Area Networks)*. Такие сети предназначены для обслуживания территории крупного города – *мегаполиса*. В то время как локальные сети наилучшим образом подходят для разделения ресурсов на коротких расстояниях и широковещательных передач, а глобальные сети обеспечивают работу на больших расстояниях, но с ограниченной скоростью и небогатым набором услуг, сети мегаполисов занимают некоторое промежуточное положение. Эти сети первоначально были разработаны для передачи данных, но сейчас они поддерживают и такие услуги как видеоконференции и интегральная передач голоса и текста.

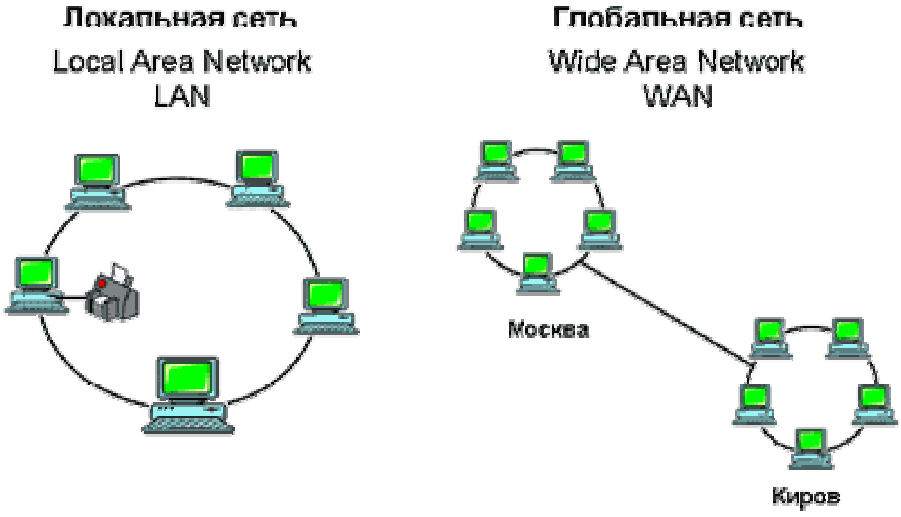


Рис. II.1.1. Локальная сеть, глобальная сеть

1.2.2. Классификация компьютерных сетей по скорости передачи данных

По скорости передачи информации компьютерные сети делятся на низко-, средне- и высокоскоростные:

– *низкоскоростные сети* – до 10 Мбит/с;

– *среднескоростные сети* – до 100 Мбит/с;

– *высокоскоростные сети* – свыше 100 Мбит/с.

1.2.3. Классификация компьютерных сетей по топологии

Существует бесконечное число способов соединения компьютеров.

Топология сети – это геометрическая форма и физическое расположение компьютеров по отношению друг к другу.

Топология определяет требования к оборудованию, тип используемого кабеля, методы управления обменом, надежность работы, возможность расширения сети.

Топология сети позволяет сравнивать и классифицировать различные сети. Выделяют три базовых топологии:

– *звезда (star)*;

– *кольцо (ring)*;

– *шина (bus).*

Шинная топология. При построении сети по шинной схеме каждый компьютер присоединяется к общему кабелю, на концах которого устанавливаются *терминаторы*.

Сигнал проходит по сети через все компьютеры, отражаясь от *конечных терминаторов*.

Шина проводит сигнал из одного конца сети к другому, при этом каждая рабочая станция проверяет адрес послания, и, если он совпадает с адресом рабочей станции, она его принимает. Если же адрес не совпадает, сигнал уходит по линии дальше (рис. II.1.2).

Достоинства:

– простота добавления новых узлов в сеть (это возможно даже во время работы сети);

– сеть продолжает функционировать, даже если отдельные компьютеры вышли из строя;

– недорогое сетевое оборудование за счет широкого распространения такой топологии.

Недостатки:

– сложность сетевого оборудования;

– сложность диагностики неисправности сетевого оборудования из-за того, что все адаптеры включены параллельно;

– обрыв кабеля влечет за собой выход из строя всей сети;

– ограничение на максимальную длину линий связи из-за того, что сигналы при передаче ослабляются и никак не восстанавливаются.

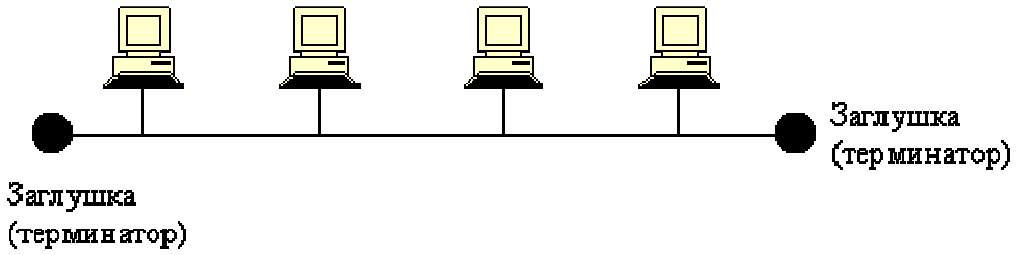


Рис. II.1.2. Шинная топология

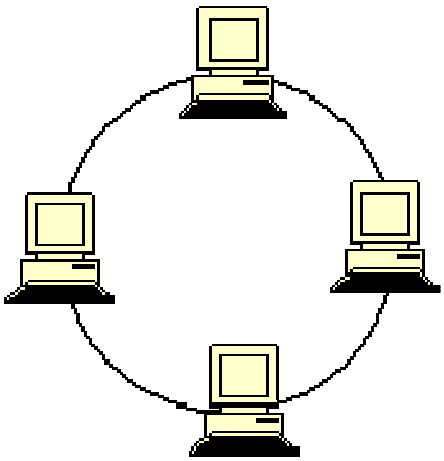


Рис. II.1.3. Топология «Кольцо»

Топология «Кольцо». Эта топология представляет собой последовательное соединение компьютеров, когда последний компьютер соединен с первым. Сигнал проходит по кольцу от компьютера к компьютеру в одном направлении. Каждый компьютер работает как *повторитель* (т.е. каждый компьютер восстанавливает приходящий к нему сигнал), усиливая сигнал и передавая его дальше. Поскольку сигнал проходит через каждый компьютер, сбой одного из них приводит к нарушению работы всей сети (рис. II.1.3).

Достоинства:

– легко подключить новые узлы, хотя для этого нужно приостановить работу сети;

– большое количество узлов, которое можно подключить к сети (более 1000);

– высокая устойчивость к перегрузкам.

Недостатки:

– выход из строя хотя бы одного компьютера нарушает работу сети;

– обрыв кабеля хотя бы в одном месте нарушает работу сети.

Топология «Звезда». Топология «Звезда» – предусматривает подключение каждого компьютера отдельным кабелем к общему устройству, называемому *концентратором (hub)*, который находится в центре сети. Концентратор служит для перенаправления передаваемой информации к одному или всем остальным компьютерам сети (рис. II.1.4).

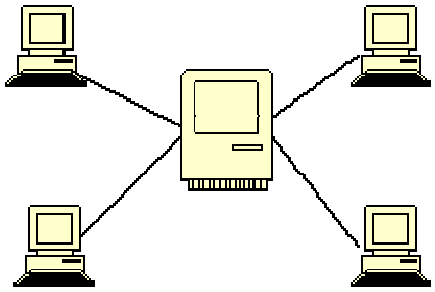


Рис. II.1.4. Топология «Звезда»

Достоинства:

– выход из строя периферийного компьютера никак не отражается на функционировании оставшейся части сети;

– простота используемого сетевого оборудования;

– все точки подключения собраны в одном месте, что позволяет легко контролировать работу сети, локализовать неисправности сети путем отключения от центра тех или иных периферийных устройств;

– не происходит затухания сигналов.

Недостатки:

– выход из строя концентратора делает сеть полностью неработоспособной;

– жесткое ограничение количества периферийных компьютеров (максимум 1024 на один концентратор);

– значительный расход кабеля.

Отметим, что по описанным базовым топологиям строятся, как правило, небольшие сети. Для крупных сетей характерно наличие произвольных связей между компьютерами, где можно, однако, выделить описанные выше топологии. Такие сети называются сетями со *смешанной топологией*. Например, *дерево (tree)* – комбинация нескольких звезд.

Каждый компьютер, который функционирует в локальной сети, должен иметь *сетевой адаптер (сетевую карту)*. Функцией сетевого адаптера является передача и прием сигналов, распространяемых по кабелям связи.

При конструировании сетей используют следующие виды кабелей:

– *неэкранированная витая пара*. Максимальное расстояние, на котором могут быть расположены компьютеры, соединенные этим кабелем, достигает 90 м. Скорость передачи информации – от 10 до 155 Мбит/с;

– *экранированная витая пара*. Скорость передачи информации – 16 Мбит/с на расстояние до 300 м;

– *коаксиальный кабель*. Отличается более высокой механической прочностью, помехозащищённостью и позволяет передавать информацию на расстояние до 2000 м со скоростью 2 - 44 Мбит/с;

– *волоконно-оптический кабель*. Идеальная передающая среда, не подвержен действию электромагнитных полей, позволяет передавать информацию на расстояние до 10 000 м со скоростью до 10 Гбит/с.

1.2.4. Классификация компьютерных сетей по способу управления

В зависимости от способа управления, сети делят на *одноранговые* и с *выделенным сервером* (*иерархические сети*).

Все компьютеры одноранговой сети равноправны. Любой пользователь сети может получить доступ к данным, хранящимся на любом компьютере.

Главное достоинство одноранговых сетей – это простота установки и эксплуатации. Главный недостаток состоит в том, что в условиях одноранговых сетей затруднено решение вопросов защиты информации. Поэтому такой способ организации сети используется для сетей с небольшим количеством компьютеров и там, где вопрос защиты данных не является принципиальным.

В иерархической сети при установке сети заранее выделяются один или несколько *серверов* – компьютеров, управляющих обменом данных по сети и распределением ресурсов. Любой компьютер, имеющий доступ к услугам сервера, называют *клиентом* сети или *рабочей станцией*.

Сервер в иерархических сетях – это *постоянное хранилище разделяемых ресурсов*. Сам сервер может быть клиентом только сервера более высокого уровня иерархии. Серверы обычно представляют собой высокопроизводительные компьютеры, возможно, с несколькими параллельно работающими процессорами, винчестерами большой емкости и высокоскоростной *сетевой картой*.

Иерархическая модель сети является более предпочтительной, так как позволяет создать наиболее устойчивую структуру сети и более рационально распределить ресурсы. Также достоинством иерархической сети является более высокий уровень защиты данных. К недостаткам иерархической сети, по сравнению с одноранговыми сетями, относятся:

– необходимость дополнительной операционной системы для сервера;
– более высокая сложность установки и модернизации сети;
– необходимость выделения отдельного компьютера в качестве сервера.

По технологии использования сервера различают сети с архитектурой *файл-сервер* и сети с архитектурой *клиент-сервер*. В первой модели используется файловый сервер, на котором хранится большинство программ и данных. По требованию пользователя ему пересылаются необходимая программа и данные. Обработка информации выполняется на рабочей станции.

В системах с архитектурой *клиент-сервер* обмен данными осуществляется между приложением-клиентом и приложением-сервером. Хранение данных и их обработка производится на мощном сервере, который выполняет также контроль доступа к ресурсам и данным. Рабочая станция получает только результаты запроса.

1.2.5. Классификация компьютерных сетей по методу коммутации

Как правило, в сетях общего доступа невозможно предоставить каждой паре абонентов собственную физическую линию связи, которой они могли бы монопольно «владеть» и использовать в любое время. Поэтому в сети всегда применяется какой-либо способ *коммутации* (соединения) абонентов, который обеспечивает разделение имеющихся физических каналов между несколькими сеансами связи и между абонентами сети.

Существуют три различные схемы коммутации в сетях:

– *коммутация каналов*;

– *коммутация пакетов*;

– *коммутация сообщений*.

Сети с *коммутацией каналов* исторически появились первыми в виде телефонных сетей. Коммутация каналов подразумевает образование составного канала из последовательно соединенных отдельных канальных участков для прямой передачи данных между узлами сети. В сети с коммутацией каналов перед передачей данных всегда необходимо выполнить процедуру установления соединения.

Указанные сети обладают высокой надежностью, минимальными затратами на маршрутизацию. К недостаткам можно отнести невозможность объединения в один канал узлов, работающих с разной скоростью передачи данных, что вызывает пульсацию скорости передачи данных

Пример. 1.1. Если сеть, изображенная на рисунке II.1.5, работает по технологии коммутации каналов, то узел 1, чтобы передать данные узлу 7, сначала должен передать специальный запрос на установление соединения коммутатору *A*, указав адрес назначения 7. Коммутатор *A* должен выбрать маршрут образования составного канала, а затем передать запрос следующему коммутатору, в данном случае *E*. Затем коммутатор *E* передает запрос коммутатору *F*, а тот, в свою очередь, передает запрос узлу 7. Если узел 7 принимает запрос на установление соединения, он направляет по уже установленному каналу ответ исходному узлу, после чего составной канал считается скоммутированным, и узлы 1 и 7 могут обмениваться по нему данными.

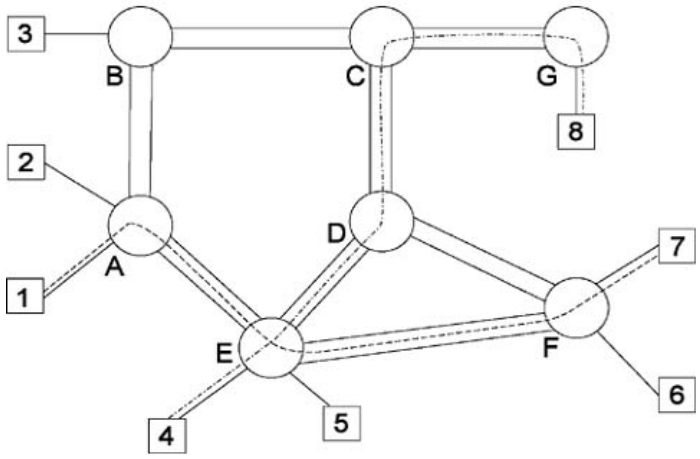


Рис. II.1.5. Коммутация каналов

Технология *коммутации пакетов* была специально разработана для эффективной передачи компьютерного трафика в сетях, где различные компьютеры могут иметь различное быстродействие.

При коммутации пакетов все передаваемые сообщения разбиваются передающим компьютером на небольшие части (от 46 до 1500 байт), называемые *пакетами*. Каждый пакет снабжается заголовком, в котором указывается адресная информация, необходимая для доставки пакета к принимающему компьютеру, а также номер пакета, используемый для «сборки» сообщения на принимающем компьютере. Пакеты транспортируются в сети как независимые информационные блоки. Специальные устройства сети коммутаторы принимают пакеты от передающих компьютеров и на основании адресной информации передают их друг другу до конечного принимающего компьютера. За счет *буферизации* (задержки) пакета во внутренней памяти коммутатора (если требуемый участок сети занят передачей другой информации) выравнивается скорость передачи данных в сети в целом и повышается пропускная способность сети (рис. II.1.6).

Под *коммутацией сообщений* понимается передача единого блока данных через промежуточные транзитные компьютеры с временной буферизацией этого блока на диске каждого компьютера. Сообщение хранится в транзитном компьютере на диске, причем время хранения может быть достаточно большим. По такой схеме обычно передаются сообщения, не требующие немедленного ответа, чаще всего сообщения электронной почты. Режим коммутации сообщений разгружает сеть для передачи сообщений, требующих быстрого ответа, например, службы WWW сети Интернет.

Техника коммутации сообщений появилась в компьютерных сетях раньше техники коммутации пакетов, но потом была вытеснена последней,

как более эффективной с точки зрения пропускной способности сети. Сегодня коммутация сообщений работает как служба прикладного уровня только для некоторых неоперативных служб.

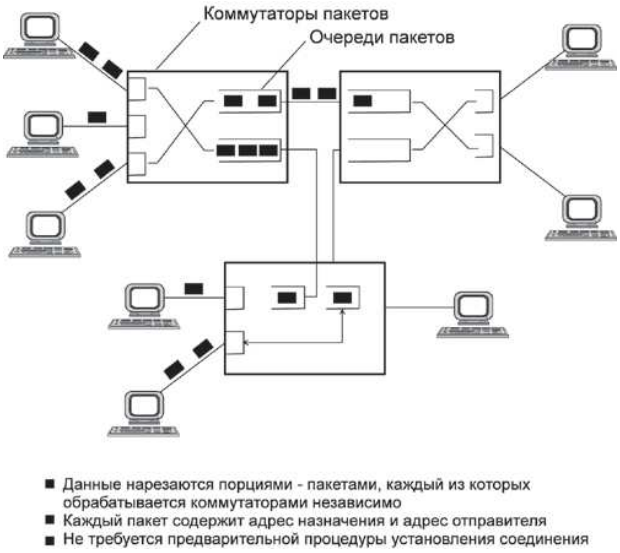


Рис. II.1.6. Коммутация пакетов

1.3. Стандартизация компьютерных сетей. Понятия интерфейса, протокола и стека

По своей сути компьютерная сеть является совокупностью компьютеров и сетевого оборудования, соединенных каналами связи. Поскольку компьютеры и сетевое оборудование могут быть разных производителей, то возникает проблема их совместимости. Без принятия всеми производителями общепринятых правил построения оборудования создание компьютерной сети было бы невозможно. Поэтому разработка и создание компьютерных сетей может происходить только в рамках утвержденных *стандартов.*

В основу стандартизации компьютерных сетей положен принцип *декомпозиции*, т.е. разделения сложных задач на отдельные более простые подзадачи. Каждая подзадача имеет четко определенные функции и строго установленные связи между подзадачами. При более внимательном рассмотрении работы компьютера в сети можно выделить две основные подзадачи:

– взаимодействие программного обеспечения пользователя с физическим каналом связи (посредством сетевой карты) в пределах одного компьютера;

– взаимодействие компьютера через канал связи с другим компьютером.

Современное программное обеспечение компьютера имеет многоуровневую модульную структуру, т.е. программный код, написанный программистом и видимый на экране монитора (модуль верхнего уровня), который проходит несколько уровней обработки, прежде чем превратится в электрический сигнал (модуль нижнего уровня), передаваемый в канал связи.

При взаимодействии компьютеров через канал связи оба компьютера должны выполнять ряд соглашений. Например, они должны согласовать величину и форму электрических сигналов, длину сообщений, методы контроля достоверности и т.д. Соглашения должны быть такими, чтобы они были поняты каждым модулем на соответствующем уровне каждого компьютера.

Суть работы многоуровневого протокола можно пояснить как «письмо в конверте». Каждый уровень протокола надписывает на «конверте» свою информацию. Сетям нужно только понимать «надпись» на «конверте», чтобы передать его в место назначения, а содержание письма не имеет значения.

На рисунке II.1.7 схематически показана модель взаимодействия двух компьютеров в сети. Для упрощения показаны четыре уровня модулей для каждого компьютера. Процедура взаимодействие каждого уровня этих компьютеров может быть описана в виде набора правил взаимодействия каждой пары модулей соответствующих уровней.

Формализованные правила, определяющие последовательность и формат сообщений, которыми обмениваются модули, лежащие на одном уровне, но в различных компьютерах называются *протоколами*.

Модули, реализующие протоколы соседнего уровня и находящиеся в одном компьютере, также взаимодействуют друг с другом в соответствии с четко определенными правилами и с помощью стандартизованных форматов сообщений. Эти правила называются *интерфейсом* и определяют набор сервисов, предоставляемых данным уровнем соседнему уровню. Другими словами, в сетевых технологиях традиционно принято, что протоколы определяют правила взаимодействия модулей одного уровня, но в разных компьютерах, а интерфейсы – соседних уровней в одном компьютере. Модули, таким образом, должны обрабатывать: во-первых, свой собственный протокол, а во-вторых, интерфейсы с соседними уровнями.

Иерархически организованный набор протоколов для взаимодействия компьютеров в сети называется *стеком коммуникационных протоколов*.

Коммуникационные протоколы могут быть реализованы как программно, так и аппаратно. Протоколы нижних уровней, как правило, реа-

лизуются комбинацией программно-аппаратных средств, а протоколы верхних уровней – только программными средствами.

Отметим, что протоколы каждого уровня обладают независимостью друг от друга, т.е. протокол любого уровня может быть изменен, не оказывая при этом никакого влияния на протокол другого уровня. Главное, чтобы интерфейсы между уровнями обеспечивали необходимые связи между ними.

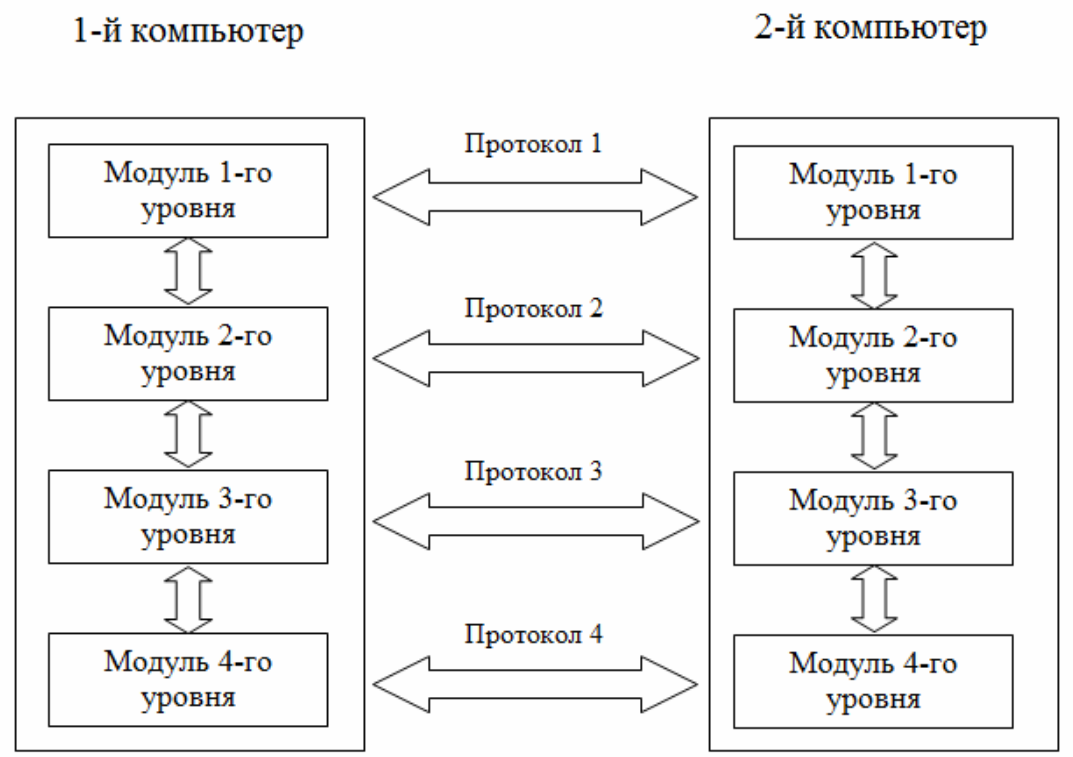


Рис. II.1.7. Взаимодействие двух компьютеров в сети

Пример. 1.2. Принцип взаимодействия компьютеров в сети можно объяснить на примере сотрудничества двух фирм. Два генеральных менеджера каждой из фирм осуществляют сделки между собой на основании заключенных договоров и соглашений. Указанные взаимодействия являются «протоколом уровня генеральных менеджеров». На каждой из фирм у менеджеров есть секретари, причем каждый менеджер имеет свой метод и стиль работы с секретарем. Один, например, предпочитает устные указания, а второй дает только письменные распоряжения. Таким образом, каждая фирма имеет свой собственный интерфейс «главный менеджер-секретарь», что не мешает, однако, нормально работать генеральным менеджерам между собой. Секретари в свою очередь договорились обмениваться информацией с помощью факсов, реализуя протокол «секретарь-секретарь». В случае если секретари перейдут на электронную почту, то генеральные менеджеры этого даже и не заметят – главное, чтобы секретари выполняли их распоряжения, т.е. должен безукоризненно работать интерфейс «менеджер-секретарь». С другой стороны, менеджеры могут заключить совершенно новый договор, т.е. изменить «протокол уровня генеральных менеджеров». Передача не старого, а нового договора на уровне секретарей пройдет для этих секретарей абсолютно не замеченной.

В рассмотренном примере мы определили два уровня протоколов – «уровень генеральных менеджеров» и «уровень секретарей». Каждый из указанных уровней имеет свой собственный протокол, который может быть изменен независимо от протокола другого уровня. Такую независимость обеспечивает правильное функционирование интерфейсов «менеджерсекретарь».

Независимость протоколов каждого уровня друг от друга и взаимодействие самих уровней посредством интерфейсов является важнейшей предпосылкой для создания ряда стандартных протоколов для компьютерных сетей.

1.4. Протокол TCP/IP

Как уже указывалось выше, основой сети Интернет является *стек проколов TCP/IP (Transmission Control Protocol/Internet Protocol)*. В этом протоколе существую четыре уровня взаимодействия:

– *канальный + физический уровень*,
– *уровень IP (адресация пакетов)*,
– *TCP (управление передачей)*,
– *прикладной уровень*.

Рассмотрим в самом общем виде принцип взаимодействия компьютеров в сети, основываясь на описанной выше многоуровневой модели.

Взаимодействие компьютеров в сети начинается с того, что приложение (программа пользователя) одного компьютера обращается к прикладному уровню другого компьютера, например, к файловой системе. Приложение первого компьютера формирует с помощью операционной системы блок данных стандартного формата, состоящее из *заголовка* и *поля данных*.

Заголовок содержит служебную информацию, которую необходимо предать через сеть прикладному уровню другого компьютера, чтобы сообщить ему, какую работу необходимо выполнить. Кроме этого в заголовке имеется информация для следующего нижнего уровня, чтобы он «знал», что делать с этим сообщением. В поле данных находится информация, которую необходимо поместить в найденный файл. Сформировав сообщение, прикладной уровень направляет его «вниз» уровню TCP. Прочитав заголовок, TCP-уровень выполняет требуемые действия над сообщением и добавляет к сообщению собственную служебную информацию – заголовок TCP-уровня, в котором содержатся указания для протоколов TCP-уровня второго компьютера. Полученное в результате сообщение передается вниз IP-уровню, который в свою очередь добавляет свой заголовок и т.д. При достижении сообщением нижнего, физического уровня, у него имеется множество заголовков, добавленных на каждом предыдущем уровне (сообщение вложено внутрь, как в матрешку). В таком виде оно и передается по сети (рис. II.1.8).

Второй компьютер принимает его на физическом уровне и последовательно перемещает его вверх с уровня на уровень (рис. II.1.9). Каждый уровень анализирует и обрабатывает заголовок своего уровня, выполняя соответствующие этому уровню функции, а затем удаляет этот заголовок и передает сообщение дальше вышележащему уровню.

Как видно из рисунка II.1.8, информация, передающаяся в линию связи, содержит большое количество служебных заголовков, которые по величине могут превосходить даже собственно данные. В результате взаимодействия протоколов всех уровней и их единому стандарту на прикладном уровне второго компьютера получаются данные, переданные первым компьютером.

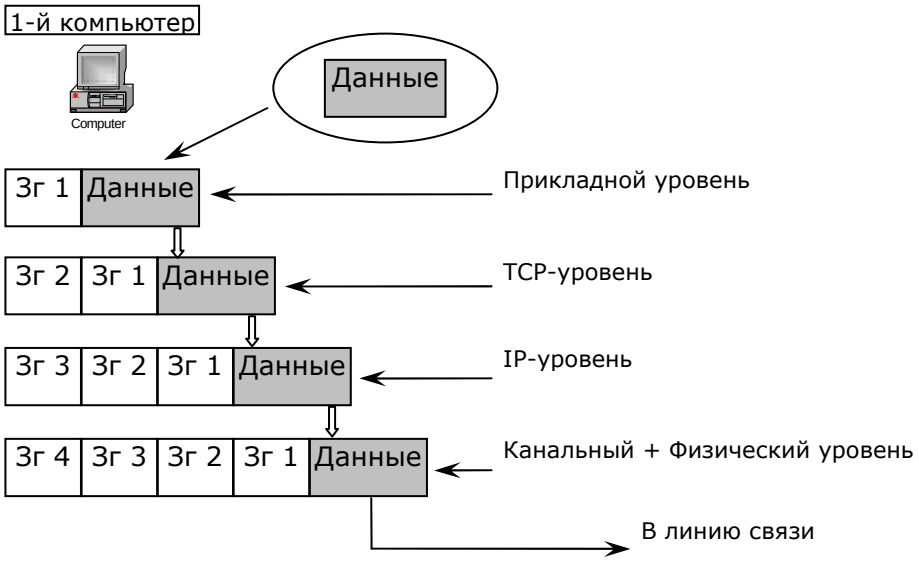


Рис. II.1.8. Передача сообщения в сеть

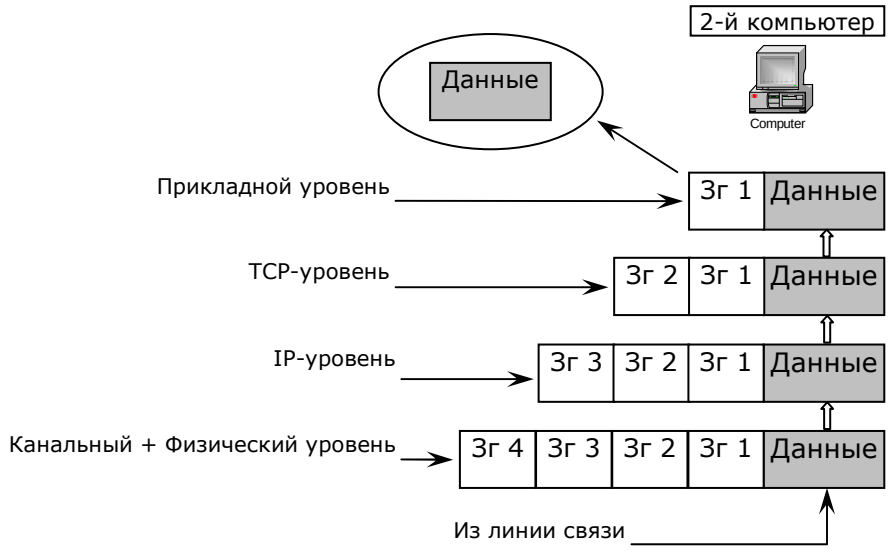


Рис. II.1.9. Прием сообщения из сети

Каковы же функции протокола TCP? Основной задачей TCP-уровня является доставка всей информации компьютеру получателя, контроль последовательности предаваемой информации, повторная отправка недоставленных пакетов в случае сбоев работы сети.

Надежность доставки информации достигается следующим образом:

– в заголовке TCP-уровня содержится *порядковый номер блока данных (сегмента),* полученных от прикладного уровня, а также специальный идентификатор, который называется *портом* (принимающий компьютер должен «знать, что ему пересылают»). Назначение идентификаторов портов осуществляется либо централизовано, если прикладные программы являются популярными и общедоступными (например, служба удаленного доступа к файлам FTP имеет порт 21, а служба WWW – порт 80), или локально – если разработчик своего приложения просто связывает с этим приложением любой доступный, произвольно выбранный номер. Этот набор идентифицирующих параметров процесса носит название *сокет.* Для каждого отправленного в сеть сегмента предающий компьютер ожидает прихода от принимающего компьютера специального сообщения – *квитанции*, подтверждающей тот факт, что компьютер нужный сегмент принял. Время ожидания прихода соответствующей квитанции называется *временем тайм-аута.* Переданный сегмент хранится в буфере на все время ожидания квитанции. В случае получения квитанции о правильности приема, TCP передает следующий сегмент, удаляя переданный из буфера, а в случае отсутствия квитанции о подтверждении приема, TCP повторяет передачу сегмента.

Для ускорения передачи сегментов в протоколе TCP организован принцип их передачи, который называется принцип «скользящего окна». Этот принцип основывается на возможности передачи нескольких сегментов в пределах одного «окна», не дожидаясь прихода квитанции на первый отправленный сегмент. На принимающем компьютере TCP, получая от уровня межсетевого взаимодействия сегменты, собирает их в блок по номерам и передает этот блок на верхний уровень приложений, отправляя обратно в сети квитанции о правильности принятого сегмента. Для производительности сети очень важным является установление времени тайм-аута и размера «скользящего окна». В общем случае для их выбора необходимо учитывать пропускную способность физических линий связи, отметим, однако, что в протоколе TCP предусмотрен специальный автоматический алгоритм определения этих величин.

На IP-уровне происходит адресация компьютеров в сети Интернет. Адрес отправителя и адрес получателя помещается в заголовок пакета. IP-адресация построена на концепции сети, состоящей из *хостов*. Х*ост* представляет собой объект сети, который может передавать и принимать IP-пакеты, например, компьютер, рабочая станция или специальное устройство – *маршрутизатор*. Хосты соединяются между собой через одну или несколько сетей. IP-адрес любого из хостов состоит *из адреса (номера) сети и адреса хоста в этой сети.*

В соответствии с принятым в момент разработки IP-протокола соглашением, адрес представляется четырьмя десятичными числами, разделенными точками. Например, сеть имеет адрес 10.1.1.10. Каждое из этих чисел не может превышать 255 и представляет один байт 4-байтного IP-адреса. Выделение всего лишь четырех байт для адресации всей сети Интернет связано с тем, что в то время массового распространения локальных сетей пока не предвиделось. Персональных компьютеров и рабочих станций не существовало. В результате под IP-адрес было отведено 32 бита.

IP-адрес назначается администратором сети во время конфигурирования компьютеров и маршрутизаторов. Номер сети может быть выбран администратором произвольным образом, или назначен по рекомендации специального подразделения Интернета – *InterNIC*. Обычно поставщики Интернет-услуг получают диапазоны адресов у подразделений InterNIC, а затем распределяют их среди своих абонентов. Отметим, что маршрутизатор может входить сразу в несколько сетей, поэтому каждый порт маршрутизатора имеет свой IP-адрес. Таким же образом и конечный компьютер может входить в несколько сетей, а значит иметь несколько IP-адресов. Следовательно, IP-адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение.

IP-адресация определяет пять классов сетей: *A, B, C, D, E*.

Сети класса *A* предназначены главным образом для использования крупными организациями. Для номеров сети выделяются 8 бит, для номеров хостов – 24 бита. Все адреса сетей класса *A* начинаются с 0 в двоичной записи или с 1 в десятичной записи, они имеют номера от 1 до 126 (если все семь бит равны «1» = 1111111= 127, номер сети 0 не используется, а номер 127 используется для специальных целей). В сетях класса *A* предусмотрено большое количество хостов – 2^{24} = 16 777 216 узлов.

Сети класса *B*. В них выделяют 16 бит для номера сети и 16 бит для номеров хостов, их адрес начинается с 10 в двоичной записи или со 128

в десятичной записи, они имеют номера от 128.0 до 191.255 (10000000.00000000 = 128.0, 10111111.11111111 = 191.255). Сети класса *B* представляют хороший компромисс между адресным пространством номера сети и номерами хостов; являются сетями среднего размера с максимальным числом хостов $2^{16} = 65\,536$.

Сети класса *C* выделяют 24 бита для номера сети и 8 бит для номеров хостов, их адрес начинается с 110 в двоичной записи или со 192 в десятичной записи, они имеют номера от 192.0.0 до 223.255.255 (11000000.00000000.00000000 = 192.0.0, 11011111.11111111.11111111 = = 223.255.255). Сети класса *C* являются наиболее распространенными сетями, число хостов в одной сети равно $2^8 = 256$.

Сети класса *D*. Адреса сетей начинаются с 1110 в двоичной записи, или с 224 в десятичной записи, они имеют номера от 224.0.0.0 до 239.255.255.255 (11100000.00000000.00000000.00000000 = 224.0.0.0, 111011111.11111111.11111111.1111111 = 239.255.255.255)

Если в пакете указан адрес сети класса *D*, то его получат все узлы этой сети. Поэтому сети класса *D* называются сетями *multicast* – широковещательными сетями и используются для обращения к группам узлов. Основное назначение *multicast* – распространение информации по схеме «один-ко-многим». Групповая адресация предназначена для экономичного распространения в Интернете или большой корпоративной сети аудио- или видеопрограмм, предназначенных большой аудитории слушателей или зрителей.

Сети класса *E*. Адреса сетей класса *E* начинаются с 11110 в двоичной записи или с 240 в десятичной записи, они имеют номера от 240.0.0.0 до 247.255.255.255 (11110000.00000000.00000000.00000000 = 240.0.0.0, 111101111.11111111.11111111.1111111 = 247.255.255.255). Сети класса *E* зарезервированы для будущих использований.

Основной недостаток использования классов IP-адресов напрямую состоит в том, что если организация имеет несколько сетевых номеров, то все компьютеры вне сети имеют доступ к этим адресам и сеть организации становится прозрачной.

Для устранения указанного недостатка адресное пространство сети разбивается на более мелкие непересекающиеся пространства – *подсети* (*subnet*) с помощью специальных кодов, которые называются *маски*. Маски также используются для увеличения адресного пространства IP-сетей. С каждой из подсетей можно работать как с обычной *TCP/IP-сетью*.

2. Глобальная сеть Интернет

2.1. Основные понятия. История развития Интернета

2.1.1. Определение

Интерне́т (произносится как [интэрнэ́т]; англ. Internet, сокр. от Interconnected Networks – объединенные сети) – глобальная телекоммуникационная сеть информационных и вычислительных ресурсов. Служит физической основой для Всемирной паутины. Часто упоминается как *Всемирная сеть*, *Глобальная сеть*, либо просто *Сеть*. Представляет собой хаотичное объединение автономных систем, что не гарантирует качества связи, но обеспечивает хорошую устойчивость и независимость функционирования системы в целом от работоспособности какого-либо ее участка.

В настоящее время, когда слово «Интернет» употребляется в обиходе, чаще всего имеется в виду Всемирная паутина и доступная в ней информация, а не сама физическая сеть.

Всемирная компьютерная сеть Интернет вместе с персональными компьютерами образует технологическую основу для развития международной концепции «Всемирного информационного общества».

2.1.2. Написание

Когда слово internet написано со строчной буквы, оно обозначает объединение сетей (англ. *interconnected networks*) посредством маршрутизации пакетов данных. В этом случае не имеется в виду глобальное информационное пространство Интернет (англ. Internet). В неанглоязычной или нетехнической среде эти понятия обычно не различают.

Словарь русского языка Российской академии наук под редакцией В.В. Лопатина рекомендует написание слова с прописной буквы: *Интерне́т*. Написание со строчной буквы используется в сложных словах, таких как «интернет-портал» и «интернет-магазин».

Некоторые издания (Яндекс, «Коммерсантъ», «Наука и жизнь» и др.) считают, что собственное имя Всемирной сети уже стало нарицательным и пишут «интернет» с маленькой буквы.

Слово «Интернет» склоняется по правилам русской грамматики как существительное мужского рода, поэтому писать следует: «в Интернете», «структура Интернета».

2.1.3. Интеренет как сеть сетей

Обращаясь в Интернет, мы пользуемся услугами *Интернет-провайдера* или *ISP* (Internet Service Provider – поставщик услуг Интернета).

ISP – это организация, которая имеет собственную высокоскоростную сеть, объединенную с другими сетями по всему земному шару. Провайдер подключает к своей сети клиентов, которые становятся частью сети данного провайдера и одновременно частью всех объединенных сетей, которые и составляют Интернет. Обычно когда слово «Сеть» пишут с большой буквы, имеют в виду именно всемирную сеть Интернет. Как правило, ISP – это крупные компании, которые в нескольких населенных пунктах имеют так называемые точки присутствия (*POP – Point of Presence*) – точки, в которых расположено аппаратное обеспечение провайдера для подключения к Интернету его клиентов. Крупный провайдер может иметь десятки точек присутствия в разных городах и тысячи клиентов. Существуют также местные провайдеры, предоставляющие услуги в одном городе.

Рассмотрим подробнее, как может быть организовано подключение пользователя к сети провайдера. Для того чтобы понять схему подключения пользователя к Интернету и принцип формирования Сети, обратимся к рисунку II.2.1.

Как видно из рисунка, пользователь подключается с помощью модема по телефонной линии, дозванивается до провайдера и устанавливает связь с одним из многочисленных модемов провайдера из *модемного накопителя* (так называемого *модемного пула*). На рисунке II.2.1 оборудование провайдера показано более подробно в одной из его точек присутствия – в городе *A*. После того как пользователь подключился к своему *ISP*, он становится частью его сети. Сеть провайдера *(ISP-A)* имеет точки присутствия также в других городах: *B* и *C*. В этих городах у провайдера тоже есть клиенты, которые подключаются к его сети.

А в городах *D* и *E* действует уже другой провайдер, *ISP-B*. Очевидно, что все клиенты провайдера *ISP-A* могут взаимодействовать между собой по сети своего провайдера, а все клиенты компании *ISP-B* – по сети своего, но если при этом нет связи между сетями *ISP-A* и *ISP-B*, то клиенты компании *A* и клиенты компании *B* не могут связаться друг с другом. Для объединения своих клиентов в одну сеть компании *A* и *B* устанавливают между собой прямое соединение с помощью так называемых точек сетевого доступа (*NAP – Network Access Points*) в разных городах. В общем случае провайдеры связываются не напрямую, а через третьих провайдеров (или их цепочку). На рисунке II.2.1 мы показали сети только двух провайдеров. Аналогично формируется подключение к сетям других провайдеров, в результате чего образуется объединение множества сетей. В Интернете действуют сотни крупных провайдеров, их сети связаны через *NAP* в различных городах.

В основе объединения малых и больших сетей (которые и составляют Интернет) лежит цепь договорных соглашений. Каждый пользователь Интернета имеет договор с определенным провайдером о подключении к его сети. В свою очередь провайдеры договариваются о соединении их сетей. Это позволяет обмениваться сообщениями всем компьютерам, подключенным к Интернету.

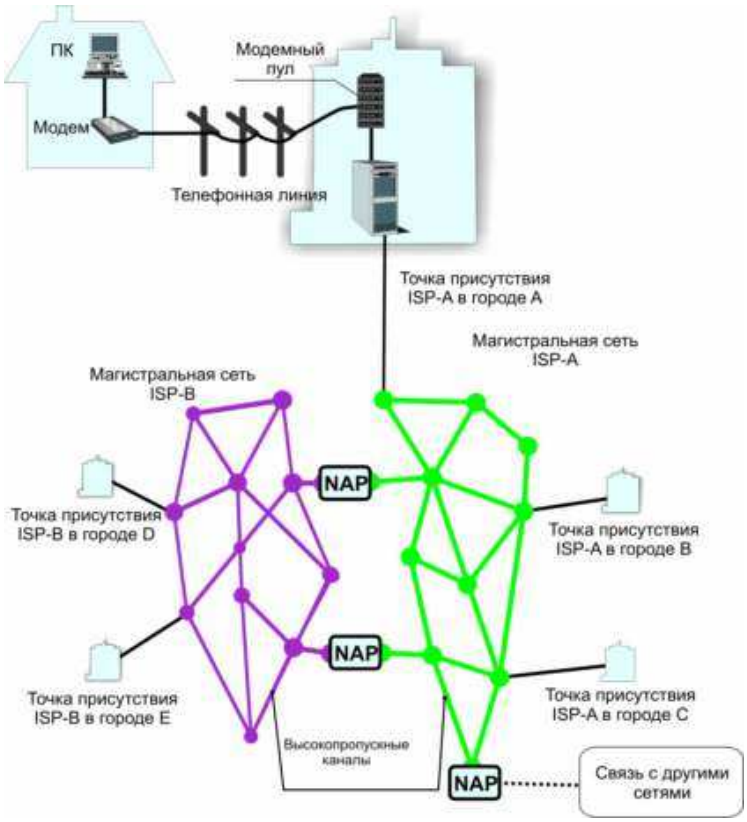


Рис. II.2.1. Обмен данных в точках сетевого доступа NAP между магистральными сетями провайдеров

2.1.4. Технология Интернета как надежная технология доставки данных

В предыдущем разделе было дано определение Интернета как глобального сообщества малых и больших сетей. Данное определение касается лишь внешней стороны явления. В более широком смысле Интернет – это информационное пространство, распределенное среди миллионов компьютеров во всем мире, которые постоянно обмениваются данными. Основная задача Интернета – это *связь*. Связь круглосуточная, высоконадежная. Для того чтобы осуществлять такую связь, была разработана специальная Интернет-технология доставки данных.

Для того чтобы кратко пояснить, в чем состоит особенность доставки информации в сети Интернет, ее можно сравнитьс телефонной сетью.

Когда абонент звонит другому абоненту по телефону в другой регион страны или даже на другой континент, система устанавливает канал между телефонами абонентов. На разных этапах сигнал может передаваться в разной среде (по медным проводам, волоконно-оптическим линиям, по радио). Но линия связи между абонентами постоянна в течение всего разговора, поэтому неполадки любого участка данной линии (например, обрыв провода) прервут разговор.

При этом если соединение нормальное, это означает, что выделенная абонентам часть сети для других уже недоступна.

Когда вы получаете на свой ПК Web-страницы с удаленного сервера, происходит совсем другой процесс.

Послание разбивается на отдельные порции данных – группы пакетов. Каждый пакет посылается на место назначения по наиболее оптимальному из доступных путей. Если какой-то пакет теряется, система посылает его заново. Поэтому, даже если какой-то участок Сети окажется нарушенным, это не повлияет на доставку пакета, который будет направлен по альтернативному пути. Таким образом, во время доставки данных между двумя пользователями нет необходимости в фиксированной линии связи.

Такую надежность сеть Интернет унаследовала от своего прототипа – сети *ARPAnet*, разработанной по заказу Министерства обороны США.

2.1.5. История развития

После запуска Советским Союзом искусственного спутника Земли в 1957 году Министерство обороны США посчитало, что на случай войны Америке нужна надежная система передачи информации. Агентство передовых оборонных исследовательских проектов США (DARPA) предложило разработать для этого компьютерную сеть. Разработка такой сети была поручена Калифорнийскому университету в Лос-Анджелесе, Стэнфордскому исследовательскому центру, Университету штата Юта и Университету штата Калифорния в Санта-Барбаре. Компьютерная сеть была названа *ARPANET* (англ. *Advanced Research Projects Agency Network*), и в 1969 году в рамках проекта сеть объединила четыре указанных научных учреждения. Все работы финансировались Министерством обороны США. Затем сеть ARPANET начала активно расти и развиваться, ее начали использовать ученые из разных областей науки.

Первый сервер ARPANET был установлен 1 сентября 1969 года в Калифорнийском университете в Лос-Анджелесе. Компьютер Honeywell DP-516 имел 24 Кб оперативной памяти.

29 октября 1969 года в 21:00 между двумя первыми узлами сети ARPANET, находящимися на расстоянии в 640 км – в Калифорнийском университете Лос-Анджелеса (UCLA) и в Стэнфордском исследовательском институте (SRI) – провели сеанс связи. Чарли Клайн (Charley Kline) пытался выполнить удаленное подключение к компьютеру в SRI. Успешную передачу каждого введенного символа его коллега Билл Дювалль (Bill Duvall) из SRI подтверждал по телефону.

В первый раз удалось отправить всего три символа «LOG», после чего сеть перестала функционировать. LOG должно было быть словом LOGON (команда входа в систему). В рабочее состояние систему вернули уже к 22:30 и следующая попытка оказалась успешной. Именно эту дату можно считать днем рождения Интернета.

К 1971 году была разработана первая программа для отправки электронной почты по сети, которая сразу стала очень популярна.

В 1973 году к сети были подключены через трансатлантический телефонный кабель первые иностранные организации из Великобритании и Норвегии, сеть стала международной.

В 1970-х годах сеть в основном использовалась для пересылки электронной почты, тогда же появились первые списки почтовой рассылки, новостные группы и доски объявлений. Однако в то время сеть еще не могла легко взаимодействовать с другими сетями, построенными на других технических стандартах. К концу 1970-х годов начали бурно развиваться протоколы передачи данных, которые были стандартизированы в 1982–83 годах. Активную роль в разработке и стандартизации сетевых протоколов играл Джон Постел. 1 января 1983 года сеть ARPANET перешла с протокола NCP на TCP/IP, который успешно применяется до сих пор для объединения (или, как еще говорят, «наслоения») сетей. Именно в 1983 году термин «Интернет» закрепился за сетью ARPANET.

В 1984 году была разработана система доменных имен (англ. Domain Name System, DNS).

В 1984 году у сети ARPANET появился серьезный соперник: *Национальный научный фонд США* (NSF) основал обширную межуниверситетскую сеть *NSFNet* (англ. National Science Foundation Network), которая была составлена из более мелких сетей (включая известные тогда сети Usenet

и Bitnet) и имела гораздо бóльшую пропускную способность, чем ARPANET. К этой сети за год подключились около 10 тыс. компьютеров, звание «Интернет» начало плавно переходить к NSFNet.

В 1988 году был разработан протокол Internet Relay Chat (IRC), благодаря чему в Интернете стало возможно общение в реальном времени (чат).

В 1989 году в Европе, в стенах Европейского совета по ядерным исследованиям (фр. *Conseil Européen pour la Recherche Nucléaire, CERN*) родилась концепция Всемирной паутины. Ее предложил знаменитый британский ученый Тим Бернерс-Ли, он же в течение двух лет разработал протокол HTTP, язык HTML и идентификаторы URI.

В 1990 году сеть ARPANET прекратила свое существование, полностью проиграв конкуренцию NSFNet. В том же году было зафиксировано первое подключение к Интернету по телефонной линии (т. н. *«дозвóн» – англ. Dialup access*).

В 1991 году Всемирная паутина стала общедоступна в Интернете, а в 1993 году появился знаменитый веб-браузер *NCSA Mosaic*.

В 1995 году NSFNet вернулась к роли исследовательской сети, маршрутизацией всего трафика Интернета теперь занимались сетевые провайдеры, а не суперкомпьютеры Национального научного фонда.

В 1995 году Всемирная паутина стала основным поставщиком информации в Интернете, обогнав по трафику протокол пересылки файлов FTP. Был образован *Консорциум всемирной паутины* (*W3C*). Можно сказать, что Всемирная паутина преобразила Интернет и создала его современный облик.

2.2. Адресация в Интернете

2.2.1. Домены

Интернет – это глобальная информационная сеть, части которой логически взаимосвязаны друг с другом посредством единого адресного пространства.

Часть в адресе электронной почты, которая находится справа от значка @, называется *доменом* и указывает на местонахождение почтового ящика.

Домен определяет имя некоторой части сети Интернет. Домены могут подразделяться на поддомены, отражающие различные области интересов или ответственности.

В доменной системе имен реализуется принцип назначения имен с определением ответственности за их подмножество со стороны соответствующих сетевых групп.

И если каждая группа убеждается, что имена, которые она присваивает, уникальны среди ее подчиненных, то никакие две системы в Сети не смогут получить одинаковые имена.

Так же уникальны адреса, которые указываются на конвертах при доставке писем обычной почтой. В мире нет стран с одинаковыми названиями. И если названия городов иногда и повторяются, то в сочетании с делением на более крупные административные единицы (районы, области) они становятся уникальными. Названия улиц не должны повторяться в пределах одного города. Таким образом, адрес на основе географических и административных названий однозначно определяет точку назначения.

Домены имеют аналогичную иерархию. Имена доменов отделяются друг от друга точками: *companya.msk.ru*, *companyb.spb.ru*.

Например, рассмотрим адрес *http://www.lab1.company.com* (рис. II.2.2).

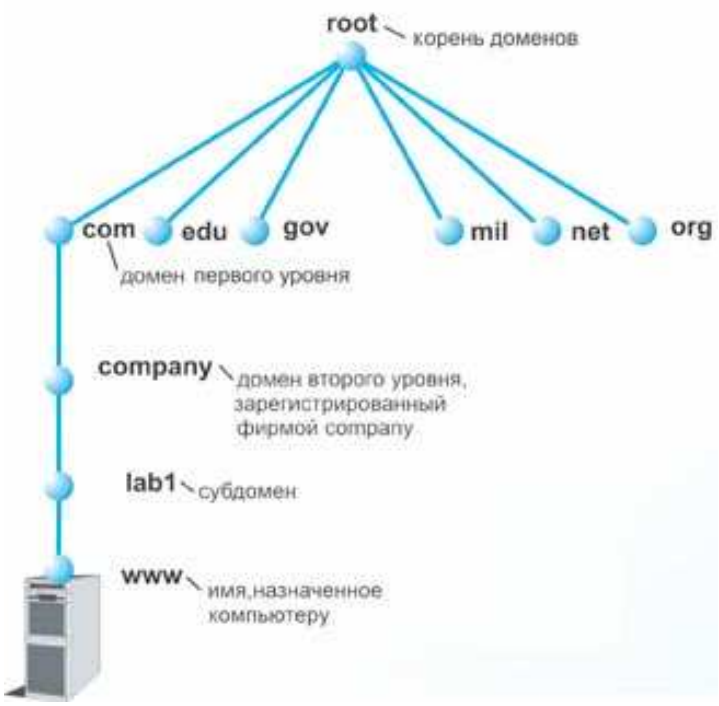


Рис. II.2.2. Структура доменных имен

Первым в имени стоит название конкретного компьютера. Это имя создано и поддерживается группой *lab1*. Группа входит в более крупное подразделение *company*, далее следует домен первого уровня – *com*.

Существуют тематические домены первого уровня и географические. Первые указывают на тип организации:

– *.com* – самое распространенное, используется в основном коммерческими структурами;

– *.edu* – относится к образовательным учреждениям;

– *.gov* – правительственные;

– *.net* – относится к различным компьютерным сетям;

– *.org* – используется в основном некоммерческими организациями;

– *.mil* – военные.

Для каждой страны есть свои географические домены: для Австралии – *au*, для Бельгии – *be* и т.д.

Внутри каждого доменного имени первого уровня находится целый ряд доменных имен второго уровня. Домен верхнего уровня располагается в имени правее, а домен нижнего уровня – левее.

На рисунке 2.3. показана структура адреса на примере российского домена.

Рассмотрим адрес *www.newcompany.yar.ru*. Домен верхнего уровня *ru* указывает на то, что адрес принадлежит российской части Интернета, *yar* – определяет город, следующий уровень – домен конкретной организации. При этом отметим, что домен второго уровня не обязательно имеет географическую привязку, как в случае *yar.ru* или *msk.ru*. На рисунке II.2.3. в качестве примера приведен домен *aha.ru*, принадлежащий компании «Зенон».

Рис. II.2.3. Структура доменных имен

2.2.2. Система адресации URL

Чтобы найти в Интернете какой-либо документ, достаточно знать ссылку на него – так называемый *универсальный указатель на ресурс* (*URL – Uniform Resource Locator*), который определяет местонахождение каждого файла, хранящегося на компьютере, подключенном к Интернету.

Адрес URL является сетевым расширением понятия полного имени ресурса в операционной системе. В URL кроме имени файла и директории, где он находится, указывается сетевое имя компьютера, на котором этот ресурс расположен, и протокол доступа к ресурсу, который можно использовать для обращения к нему.

Для того чтобы лучше разобраться с системой адресации, рассмотрим следующий URL: *http://www.lipunov.msk.ru/prochn/lab/IVANOV.htm* и его структуру.

Первая часть *http://* (*HyperText Transfer Protocol* – протокол передачи гипертекста, по которому обеспечивается доставка документа с Web-сервера Web-браузеру) указывает программе просмотра (браузеру), что для доступа к ресурсу применяется данный сетевой протокол. Схема указания способа доступа перед указанием адреса наверняка вам встречалась и прежде. Например, если на визитке вы видите подобную запись:

Тел: 433-88-15.
Факс: 433-88-14,

то очевидно, что по первому номеру возможен телефонный разговор, а по второму – отправка факса. Аналогично в URL первым стоит указатель на тип доступа к запрашиваемому файлу, а затем его адрес.

Вторая часть *www.lipunov.msk.ru* указывает на доменное имя и адресует конкретный компьютер или группу компьютеров, выполняющих одинаковую задачу.

Третья часть *prochn/lab/IVANOV.htm* показывает программе-клиенту, где на данном компьютере-сервере искать ресурс. В рассматриваемом случае ресурсом является файл в формате *html*, а именно *IVANOV.htm*, который находится в папке *lab*, которая в свою очередь расположена в папке *prochn*. При этом речь может идти не о физической папке на диске компьютера – система адресации файлов в рамках сервера может быть полностью виртуальной (т.е. создается впечатление, что папки с файлами существуют, хотя в действительности это не так).

Примечание 2.1. Внимание! При написании URL важно различать прописные и строчные буквы.

Для большей наглядности проведем аналогию с доставкой обычного письма в адрес некоторой организации (например, института) на имя конкретного человека. Как видно из рисунка II.2.4, можно установить полную аналогию в структуре адреса.

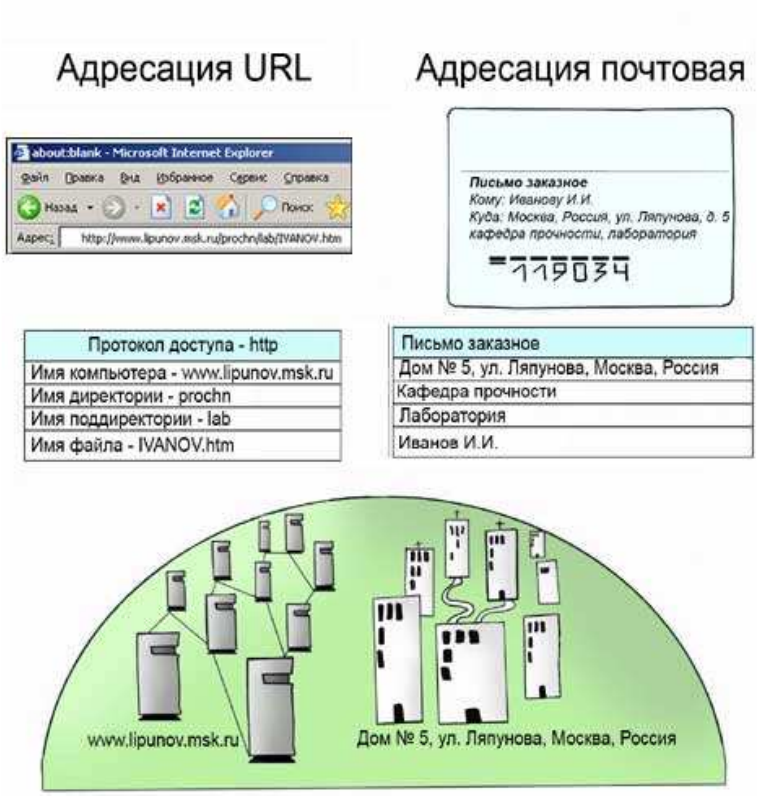


Рис. II.2.4. Система адресации URL
и адресация обычной почтовой службы

Отметим, что имена каталогов, содержащиеся в URL, обычно являются виртуальными и не имеют ничего общего с реальными именами каталогов компьютера, на котором выполняется Web-сервер, а являются их псевдонимами: ни один владелец компьютера, на котором выполняется Web-сервер, не позволит постороннему пользователю, обращающемуся к Web-серверу через Интернет, получить доступ к реальной файловой системе этого компьютера.

2.3. Обзор сервисов Интернета

Когда говорят о работе в Интернете или об использовании Интернета, то на самом деле речь идет не об Интернете в целом, а только об одной или нескольких из его многочисленных служб. В зависимости от конкретных целей и задач клиенты Сети используют те службы, которые им необходимы.

В простейшем понимании *служба* – это пара программ, взаимодействующих между собой в соответствии с определенными *протоколами*. Одна из программ этой пары называется *сервером*, а вторая – *клиентом*. Соответственно, когда говорят о работе служб Интернета, речь идет о взаимодействии серверного оборудования и программного обеспечения и клиентского оборудования и программного обеспечения.

2.3.1. Telnet

Исторически одной из ранних является служба удаленного управления компьютером *Telnet*. Подключившись к удаленному компьютеру по протоколу этой службы, можно управлять его работой. Такое управление еще называют *консольным* или *терминальным*. В прошлом эту службу широко использовали для проведения сложных расчетов на удаленных вычислительных центрах. Так, например, если для очень сложных вычислений на персональном компьютере требовались недели непрерывной работы, а на удаленной супер-ЭВМ всего несколько минут, то персональный компьютер применяли для удаленного ввода данных в ЭВМ и для приема полученных результатов.

В наши дни, в связи с быстрым увеличением мощности персональных компьютеров, необходимость в подобной услуге сократилась, но, тем не менее, службы Telnet в Интернете продолжают существовать. Часто протоколы Telnet применяют для дистанционного управления техническими объектами, например: телескопами, видеокамерами, промышленными роботами, автоматизированными складами и даже торговыми автоматами.

2.3.2. Электронная почта: адрес электронной почты и почтовые клиенты, принципы работы электронной почты. Web почта

Электронная почта. *Электронная почта* является неотъемлемой частью современного бизнеса: она обеспечивает быструю связь, увеличивает производительность труда и снижает расходы.

Работа с электронной почтой занимает более четверти времени сотрудников современной компании. Электронная почта давно обошла по популярности традиционную почту: ежегодно в мире рассылается десятки миллиардов электронных писем. Электронная почта может быть прочитана в удобное время, ее можно разослать сразу большому количеству получателей. Хранение писем в базе данных почтового клиента позволяет осуществлять быстрый поиск и сортировку почтовых отправлений. А главное, электронная почта намного быстрее и дешевле обычной почтовой рассылки.

Адрес электронной почты. В момент регистрации доступа в Интернет провайдер, как правило, предоставляет пользователю дисковое пространство под почтовый ящик, имеющий уникальный адрес (*E-mail Account Address*), а также имя пользователя (*E-mail Account Login Name*) и пароль (*E-mail Account Password*) для предотвращения несанкционированного доступа к почте.

Адрес электронной почты имеет следующий формат:

имя_пользователя@имя_домена, например, *Petrov@abc.msk.ru*.

Часть слева от значка @ – это *имя почтового ящика* (*E-mail Account Name*) на сервере, из которого владелец адреса забирает письма (в данном примере – *Petrov*). Как правило, имя пользователя совпадает с именем почтового ящика.

Часть справа от значка @ называется *доменом* и указывает на местонахождение этого почтового ящика. Следует отметить, что адрес электронной почты определяет не адрес домашнего компьютера пользователя, а адрес сервера, на котором он получает почту. Сегодня каждый пользователь может зарегистрировать несколько бесплатных адресов на разных почтовых серверах в разных частях света.

Рисунок II.2.5 иллюстрирует понятие адреса электронной почты. Некий пользователь может зарегистрировать адрес на сервере в США, например, *petya@ny-server.com*, и на сервере в Москве – *petya@moscow-server.ru*. И из каждого ящика такой пользователь может получать письма на свой домашний компьютер, копируя данные и с сервера в Нью-Йорке, и с сервера в Москве, установив переадресацию с одного *e-mail* на другой.

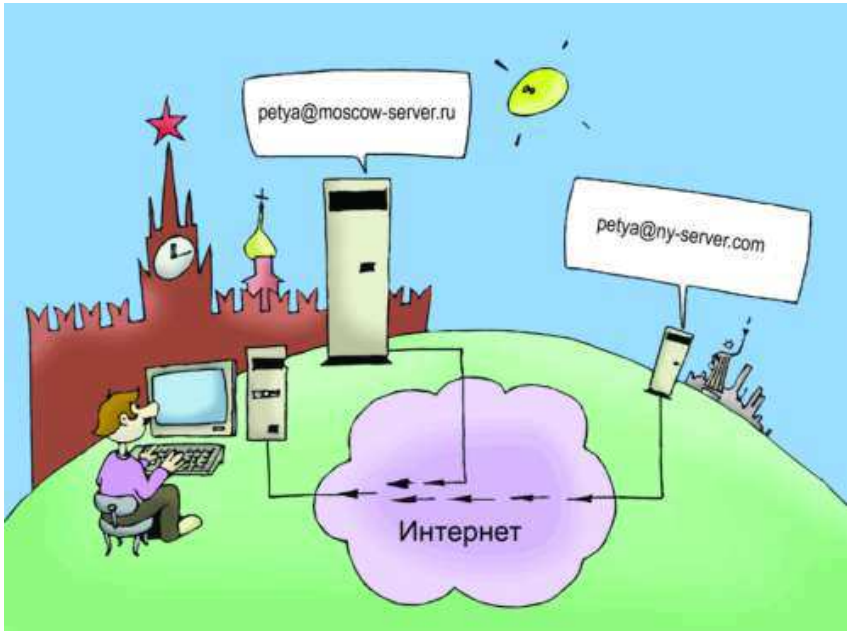


Рис. II.2.5. Связь электронного адреса с сервером,
на котором зарегистрирован его почтовый ящик

Электронная почта построена по принципу *клиент-серверной архитектуры*. Пользователь общается с *клиентской программой*, которая в свою очередь связывается с *почтовым сервером*.

Для отправления писем используются протокол *SMTP* (*Simple Mail Transfer Protocol* – простой протокол пересылки почты) и, соответственно, SMTP-серверы.

Для приема почтовых сообщений сегодня чаще всего используется протокол *POP3* (*Post Office Protocol*), который контролирует право пользователя забирать почту из ящика и поэтому требует предоставления имени пользователя и пароля. Сейчас с протоколом POP3 почти на равных конкурирует *IMAP4*, который дает возможность выборочно копировать приходящие письма с почтового сервера на компьютер. IMAP позволяет работать со своей почтой на сервере так же, как и на локальной машине. С помощью этого протокола можно работать со своим почтовым ящиком с разных компьютеров – пользоваться одним и тем же адресом электронной почты на работе, дома или в командировке.

Функции почтовых клиентов. Для работы с почтой необходима программа – *почтовый клиент*. Основные функции почтовых клиентов – это прием сообщений, обеспечение их просмотра, сортировка, автоматизация создания ответных сообщений и поддержка адресной книги.

При подготовке электронного письма пользователь готовит текст сообщения и заполняет ряд стандартных полей:

– *To (кому)*,

– *CC (копия)*,

– *BCC – Blind Carbon Copy (слепая копия)*[1].

Разница между полями *CC* и *BCC* заключается в том, что все те адреса, которые будут указаны в поле *CC*, будут видны всем участникам переписки, а указанные в поле *BCC*, не будут видны остальным адресатам.

Сведения о дате создания письма и адрес почтового ящика отправителя заполняются автоматически. Почтовая программа позволяет упростить процесс заполнения этих полей, если человек отвечает на письмо. Особенно удобны, в частности, такие функции, как *Ответить*, *Ответить всем* и *Перенаправить*. Для создания текста электронного письма обычно предоставляется текстовый редактор, который позволяет производить операции форматирования, использовать буфер обмена для копирования фрагментов из имеющихся документов, выбирать кодировку текста. Если почтовый клиент поддерживает HTML-формат, то возможности оформления письма существенно увеличиваются: можно использовать встроенные мультимедийные объекты, голосовые и видеосообщения.

[1] Сокращение от англ. – *carbon copy*, дословно «полученное под копирку».

Большинство почтовых клиентов позволяют автоматически перенаправить полученное сообщение по новому адресу. При ответе на письмо ряд полей заполняется автоматически. В частности, нет необходимости набирать адрес получателя – достаточно нажать кнопку Ответ.

Для удобства хранения и поиска почтовый клиент обычно позволяет рассортировать сообщения по логическим папкам. Распределение писем по электронным папкам позволяет заметно сэкономить время, затрачиваемое на разбор почты. Важно выработать правильную систему сортировки почты, тогда не придется искать нужное послание в «общей куче». Однако когда назначено папок так много, что приходится рназмышлять, в какую же из них могло попасть то или иное послание, это тоже замедляет работу. Как бы ни была оптимизирована сортировка почты, необходимо своевременно избавляться от старых ненужных сообщений, это поможет существенно сократить время на поиск.

Популярные почтовые клиенты. В мире существует несколько сотен почтовых клиентов. Мы упомянем лишь самые распространенные программы и укажем адреса, где их можно получить.

Outlook Express – наиболее популярная программа, ее можно бесплатно скопировать с сервера *http://www.microsoft.com/*. Интерфейс программы Outlook Express представлен на рисунке II.2.6.

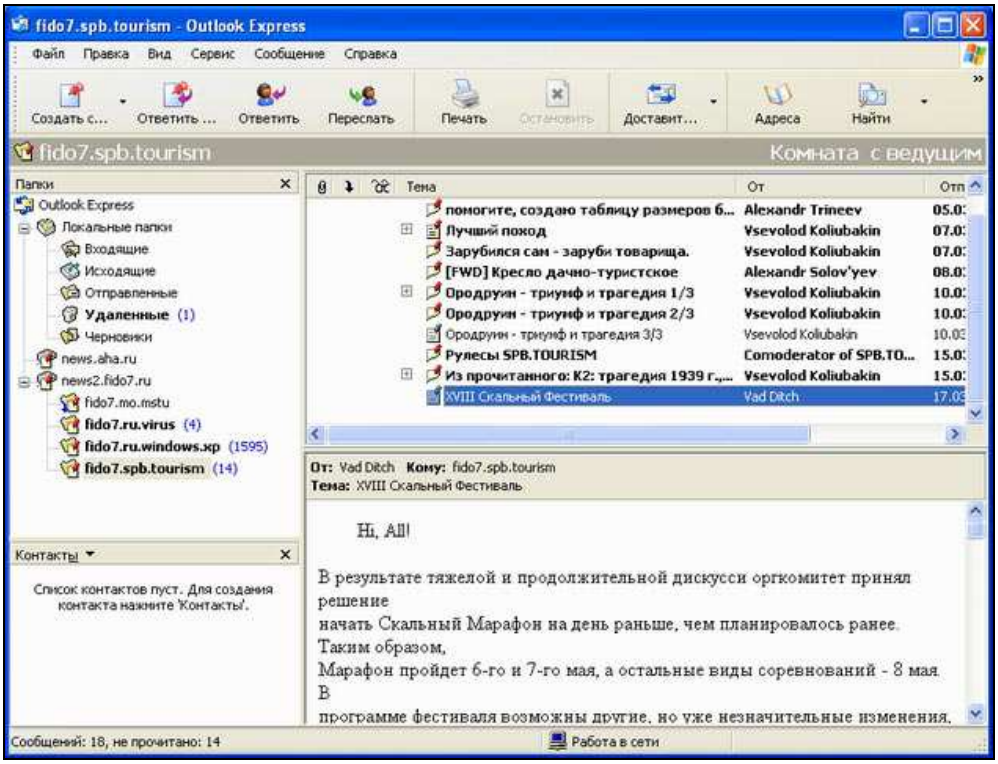


Рис. II.2.6. Интерфейс программы Outlook Express

Определенную популярность среди русскоязычных пользователей имеет программа *The Bat!* – продукт молдавских разработчиков, найти который можно по адресу *http://www.ritlabs.com/ru/the_bat/*. В программе реализована поддержка русского языка, полностью русифицирован интерфейс.

Очень удобный почтовый клиент предложила компания Microsoft в составе своего продукта *Outlook 2003*. В данной версии программа получила обновленный пользовательский интерфейс. Трехпанельный дизайн (рис. II.2.7) обеспечивает более рациональное расположение сообщений на экране.

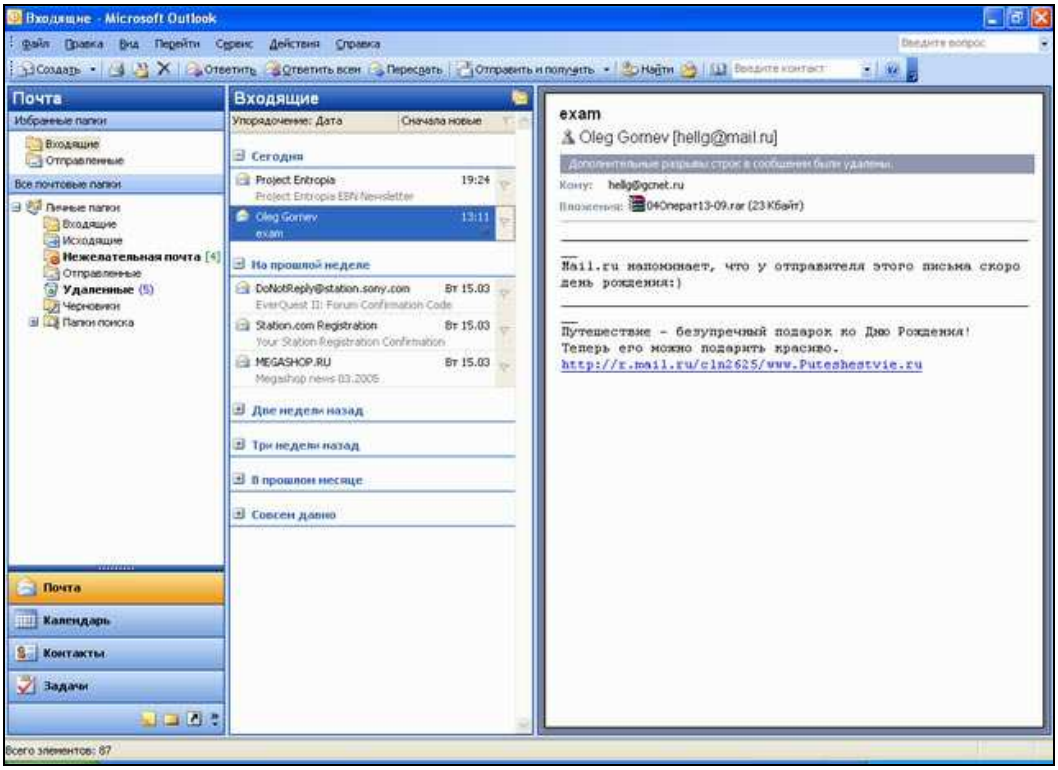


Рис. II.2.7. Обновленный интерфейс Outlook 2003

Представление области просмотра в виде вертикальной панели позволяет выделить для текста письма на 40% больше места, чем при традиционном горизонтальном расположении панели, что в свою очередь дает возможность реже открывать письма в отдельном окне и тем самым избавляет от необходимости следить за множеством отдельных окон для каждого сообщения. Следует также отметить, что вертикальное (газетное) расположение текста в виде узкой колонки удобнее для чтения (особенно при наличии большого экрана).

Как работает почта. Рассмотрим конкретный пример. Пусть некоторый владелец электронного ящика с адресом *vasya@abc.ru* на почтовом сервере *abc.ru* пишет письмо владельцу почтового ящика с адресом *masha@def.com* на сервере *def.com*.

Для того чтобы подготовить письмо, он вызывает клиентскую программу, создает текст сообщения и в графе *Кому* указывает адрес получателя *masha@def.com* (рис. II.2.8). Если отправитель не имеет постоянного подключения к Интернету, то после нажатия кнопки Отправить он устанавливает сеанс связи с провайдером и начинает получать накопившуюся почту и отправлять подготовленные письма. Порядок отправления почты зависит от текущих настроек почтовой программы. Обычно письма складываются в определенную папку и отправляются другой командой после установки связи с провайдером.

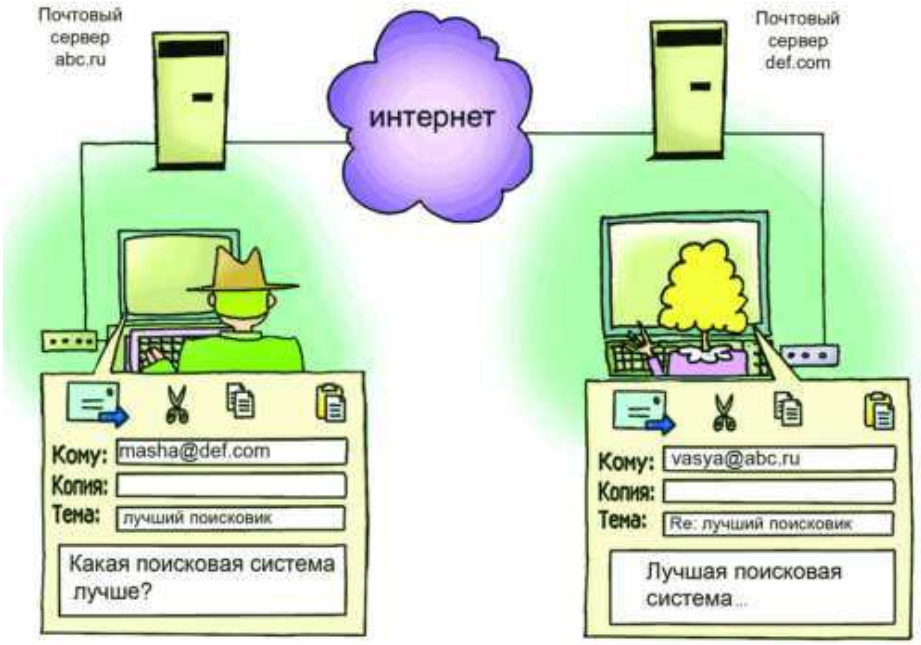


Рис. II.2.8. Общение пользователя с клиентской программой, которая взаимодействует с E-mail-сервером

После установления подключения к Интернету клиентская программа соединяется с почтовым сервером и передает ему почтовый адрес отправителя *vasya@abc.ru* и текст самого сообщения. При отправке почты клиентская программа взаимодействует с сервером исходящей почты, т.е. *SMTP-сервером*, по протоколу *SMTP*. Имя SMTP-сервера, которому передается почта, должно быть заранее введено в клиентскую почтовую программу при ее настройке.

Процедура отправки электронной почты заключается в копировании вновь подготовленных сообщений из базы клиента на SMTP-сервер (в нашем случае – сервера *abc.ru*), который связывается с почтовым сервером получателя и записывает письмо в базу его почтового ящика на этом сервере (в нашем случае – сервера *def.com*).

Web-почта. *Web-почта* – это почта с Web-интерфейсом. Для доступа к такой почте достаточно иметь доступ в Интернет. Можно даже не иметь на своем компьютере почтовых программ, работающих по протоколам POP3/SMTP (*Microsoft Outlook Express*). Достаточно и обыкновенного браузера. Mail.ru, Hotbox.ru, Yahoo.com, Hotmail.com – все это примеры популярных служб Web-почты.

Web-почта предоставляет все типичные функции электронной почты. На Web-страничке пользователю предлагается ввести имя и пароль, после чего он видит свои письма и работает с ними, почти как у себя на компьютере.

Основными достоинствами Web-почты являются доступность с любого компьютера, имеющего выход в Интернет, универсальность адреса, не зависящего от места работы, и простое имя типа *user@mail.ru*.

К недостаткам следует отнести необходимость передавать свои имя и пароль по сети за пределы своей фирмы, что снижает уровень безопасности. Кроме того, в бесплатных службах проблемы создает присутствие рекламы.

Впрочем, существуют и платные службы Web-почты. Платные службы не докучают рекламой, обычно предоставляют большой размер почтового ящика, несут более значительную ответственность за предоставление сервиса.

2.3.3. Телеконференции Usenet

Очень похожей на электронную почту службой является *служба телеконференций Usenet*. Для работы этой службы используются те же самые программы *почтовые клиенты*. Разница в том, что сообщение отсылается не одному получателю, а сразу всей *новостной группе (News Group)*. Следовательно, это сообщение прочитают все пользователи данной группы. Телеконференции позволяют обсудить какую-либо тему, и каждый может свободно выразить свое мнение, соблюдая определенный этикет.

Для работы этой службы используется протокол *NNTP (Network News Transfer Protocol)* – сетевой протокол передачи новостей.

2.3.4. World Wide Web

Определение. Основные понятия. *WWW* (*World Wide Web, Всемирная паутина*) – самый популярный сервис Интернета, который определил столь массовое обращение к ресурсам Сети.

World Wide Web – это единое *информационное пространство*, состоящее из взаимосвязанных и форматированных специальным образом документов (*HTML-документов*), хранящихся на *Web-серверах*.

Под термином Web-сервер, в зависимости от контекста, может подразумеваться как аппаратная, так и программная часть: *аппаратный Web-сервер* – это компьютер, на котором выполняется *программа Web-сервер* – приложение, получающее запросы и выполняющее определенные действия в соответствии с этими запросами. Например, оно может запускать приложения и генерировать документы.

Служба *WWW* реализована в виде клиент-серверной архитектуры. Пользователь с помощью клиентской программы (*браузера* (см. ниже «Средства просмотра Web»)) осуществляет запрос той или иной информации на сервере, а Web-сервер обслуживает запрос браузера (рис. II.2.7).

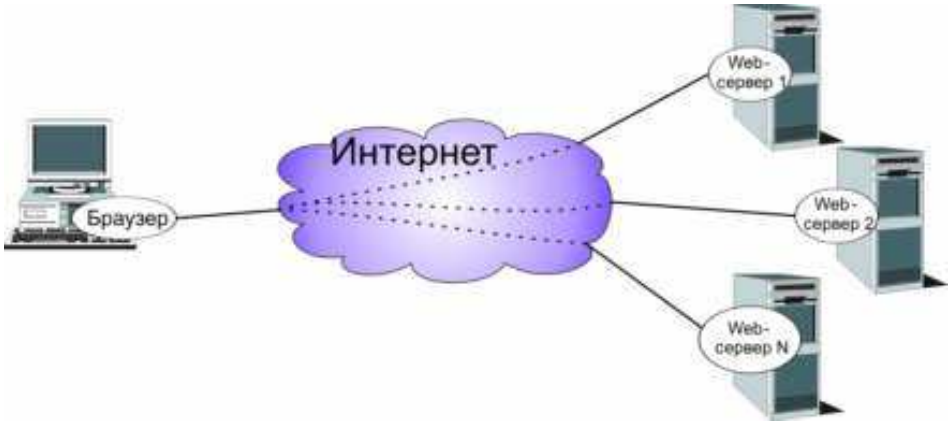


Рис. II.2.7. Схема обращения браузера к искомому ресурсу на сервере

Современный *браузер* – это программа с графическим интерфейсом, которая обеспечивает обращение к искомому ресурсу на сервере по его URL (универсальному адресу ресурса). Браузер считывает запрашиваемый документ, форматирует его для представления пользователю и демонстрирует на клиентском компьютере.

Web-страница. Отдельные документы, составляющие *пространство Web*, называют *Web-страницами*, а группы страниц, связанных общим именем, темой и объединенных навигационно, – *Web-сайтами*. Первую страницу, которую видит пользователь при обращении на тот или иной ресурс, называют стартовой, домашней или индексной страницей (*home page*). Система *гиперссылок* (см. ниже «Гиперссылки») определяет структуру Web-сайта. Страницы на сайте могут иметь *линейную*, *древовидную* структуру, но чаще на каждой странице имеется несколько ссылок, что и позволяет говорить о структуре «*паутина*» (рис. II.2.8).

Очевидно, что web-страницы, входящие в состав сайта, должны где-то физически размещаться и быть доступными любым пользователям в течение 24 часов.

Для этих целей предназначены web-сервера, которые, как правило, размещены в специально оборудованных для этого помещениях, и представляют собой обычные компьютеры, оптимизированные под выполнение данной задачи. Web-сервера имеют постоянный и широкополосный доступ в интернет для того, чтобы пользователи из любой точки земного шара и в любом количестве могли получить доступ к тому или иному сайту.

Услуги по предоставлению места для размещения сайтов предлагают так называемые *хостеры*. В зависимости от посещаемости вашего сайта вам может понадобиться как весь web-сервер целиком (услуга предоставления выделенного сервера), так и его малая часть (виртуальный хостинг или виртуальный выделенный сервер).

Любая web-страница (web-документ) сайта имеет свой собственный уникальный web-адрес или другими словами URL (Uniform Resource Locator). Для того чтобы пользователь увидел содержимое этой web-страницы у себя на экране компьютера, его браузер должен будет завязать диалог с web-сервером путем посылки *http-запросов* и получения *http-ответов*.

В результате этого диалога браузер получает *html-код* web-документа, разбирает его, подгружает все необходимые дополнительные элементы оформления web-страницы (*изображения, ccs-файлы, скрипты*), и пользователь видит на экране страницу сайта.

Рис. II.2.8. Различные варианты структуры сайта

Для того чтобы лучше понять идею организации Всемирной паутины, обратимся к условному примеру (рис. II.2.9), в котором пользователь из США ищет Web-сайт с московскими новостями, не зная адреса сервера, представляющего эту информацию. Предположим, что он набирает известный ему адрес Web-сайта, который физически расположен на компьютере в США и посвящен теме «Новости в мире». Не исключено, что на данном сервере нет искомой информации но, скорее всего, есть ссылки на новостные сайты разных регионов. Возможно выбрав ссылку «Европа», пользователь соединится уже с другим компьютером, например расположенным где-то в Европе, затем по ссылке «Россия» перейдет на компьютер в России и далее по ссылке «Москва» соединится с четвертым сервером, с помощью которого получит необходимые данные.

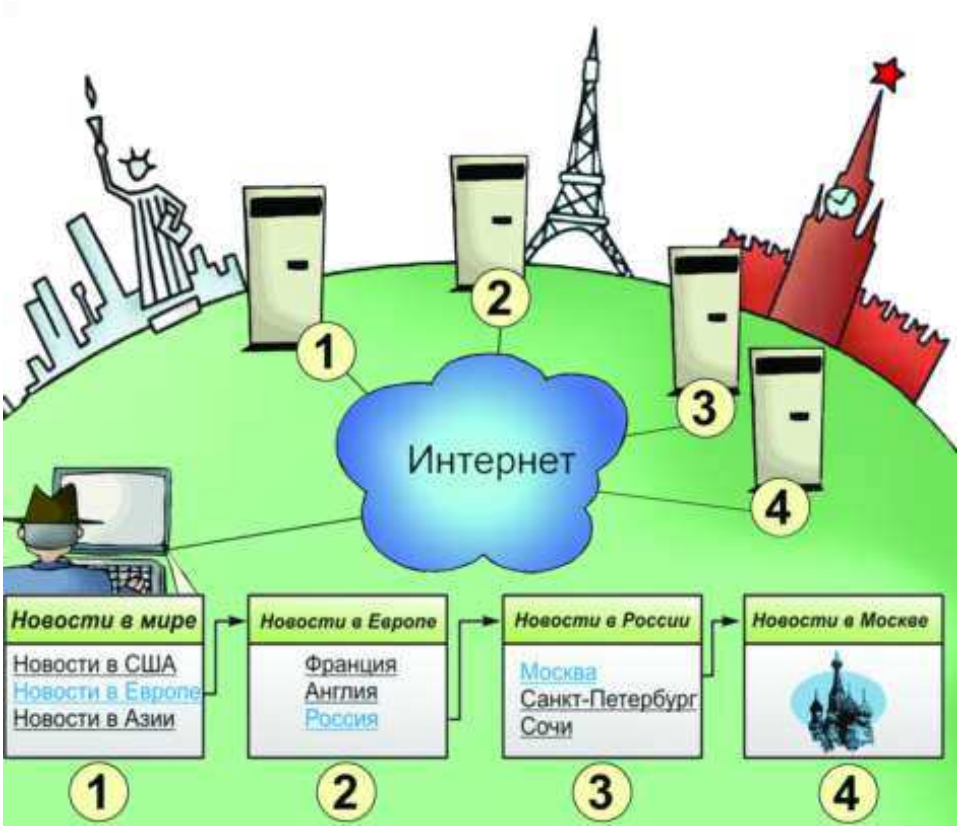


Рис. II.2.9. Принцип организации Всемирной паутины

Обычно Web-страница – это *комбинированный документ*, который может содержать текст, графические иллюстрации, мультимедийные и другие объекты. Для создания Web-страниц используется язык *HTML* (*HyperText Markup Language – язык гипертекстовой разметки*).

Гиперссылки. Отличительной особенностью среды World Wide Web является наличие средств перехода от одного документа к другому, тематически с ним связанному, без явного указания адреса. Связь между документами осуществляется при помощи *гиперссылок*.

Гиперссылка – это выделенный фрагмент документа (текст или иллюстрация), с которым ассоциирован адрес другого Web-документа. При использовании гиперссылки (обычно для этого требуется навести на нее указатель мыши и один раз щелкнуть) происходит переход по ней – открытие Web-страницы, на которую указывает ссылка. Процесс перемещения между Web-документами с помощью гипертекстовых ссылок получил название *навигации* или *серфинга*. В том, что для серфинга не требуется знать местоположение искомых документов, как раз и заключается основное удобство службы *WWW*.

Следует отметить, что идея нелинейного прочтения текста возникла задолго до появления компьютеров. Еще в Библии применялись специальные пометки на полях, отсылающие читателя к другим страницам книги. Практическое применение идеи появилось с возникновением электронных документов, задолго до появления службы WWW.

Браузеры. Современный *браузер* – это программа с графическим интерфейсом, которая обеспечивает обращение к искомому ресурсу на сервере по его URL (универсальному адресу ресурса). Браузер считывает запрашиваемый документ, форматирует его для представления пользователю и демонстрирует на клиентском компьютере.

Основные функции браузеров:

– установление связи с Web-сервером, на котором хранится документ, и загрузка всех компонентов комбинированного документа;

– интерпретация тэгов языка *HTML*, форматирование и отображение Web-страницы в соответствии с возможностями компьютера, на котором браузер работает;

– предоставление средств для отображения мультимедийных и других объектов, входящих в состав Web-страниц, а также механизма расширения, позволяющего настраивать программу на работу с новыми типами объектов;

– обеспечение автоматизации поиска Web-страниц и упрощение доступа к Web-страницам, посещавшимся ранее;

– предоставление доступа к встроенным или автономным средствам для работы с другими службами Интернета.

Кроме основных браузеры обладают дополнительными функциями, например, обеспечивают упрощение поиска, хранение закладок, указывающих на избранные страницы, и т.п.

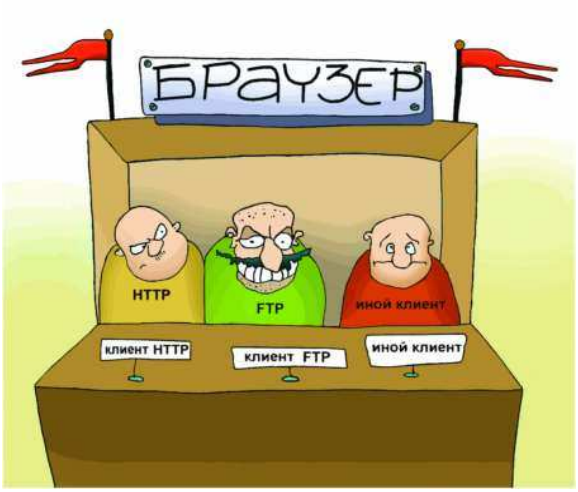


Рис. II.2.10. Современный браузер как средство интеграции нескольких клиентских программ

Созданные первоначально для просмотра HTML-документов, браузеры постепенно стали универсальными *Интернет-клиентами*. Современный браузер можно представить как приложение, в которое интегрированы несколько клиентских программ (рис. II.2.10). Браузер прочитывает URL и, в зависимости от указанного протокола, «понимает», как нужно обрабатывать данные. Например, достаточно часто браузеры используют для просмотра и передачи файлов по FTP-протоколу.

Механизм работы Web-сервера. Теперь более подробно опишем механизм работы Web-сервера и браузера. Рассмотрим пример обращения к ресурсу *www.translate.ru* (рис. 2.11).

В адресной строке браузера набираем необходимый *URL* (рис. II.2.11, пункт 1), после чего браузер получает информацию об используемом протоколе (*http*) и имени сервера (*www.translate.ru*). Браузер устанавливает связь с искомым Web-сервером и, используя протокол *HTTP*, запрашивает искомый ресурс. Сервер посылает браузеру HTML-страницу, которая хранится на сервере (рис. II.2.11, пункт 2). Обычно даже простая Web-страница содержит не только текст, но и графику, т.е. состоит из нескольких файлов разного типа. Браузер считывает HTML-тэги, воссоздает страницу на экране компьютера, и мы видим результат своего запроса (рис. II.2.11, пункт 3).

Рис. II.2.11. Схема работы Web-сервера

В данном случае мы рассмотрели пример работы так называемых *статических страниц*.

Статические страницы представляют собой точную копию файлов, лежащих в каталогах Web-сервера, и не изменяются до тех пор, пока разработчик сам в них что-то не поменяет. Однако страницы могут формироваться *динамически*, т.е. во время обработки запроса по какой-то программе, а не из готового файла на диске. Вы наверняка сталкивались со страницами, которые были созданы по вашему запросу динамически, – как говорится, «сформированы на лету». Например, любая книга отзывов на Web-сайте предоставляет определенную форму, в которую вы добавляете свой текст, когда вы в следующий раз открываете эту страницу, она содержит новое сообщение.

Аналогично, когда поисковой машине дается запрос по поводу некоторого документа, и она выдает список ссылок, очевидно, что этот список формировался именно на данный запрос, а не хранился в Сети заранее.

Кэширование Web-страниц. Просматривая Web-ресурсы, Пользователи зачастую сталкиваются с тем, что отдельные страницы загружаются значительно медленнее других. Часто документ передается с задержкой, если сервер, с которого он скачивается, имеет низкоскоростной канал выхода в Интернет или работает в режиме перегрузки. Возникает вопрос – как ускорить загрузку. Одно из решений состоит в том, чтобы не скачивать данные с удаленных серверов по нескольку раз.

При навигации в Web часто возникает необходимость, как в книге, «перелистнуть страницу назад», т.е. загрузить ее повторно. Очевидно, что когда пользователь в браузере нажимает кнопку Назад для того чтобы вернуться к просмотренной ранее статичной странице, нет смысла повторно скачивать ее с сервера. Логично сохранить просмотренный документ на диске компьютера в специальной буферной зоне (*кэше*) и быстро отобразить его, если пользователь захочет вернуться по ссылкам назад. Именно так и поступают современные браузеры.

В современных браузерах (Internet Explorer и Netscape Navigator) существует возможность настройки объема папки, в которой сохраняются кэшированные документы.

Вспомогательные программы для работы в Web. Большинство начинающих пользователей, которые обращаются к ресурсам Интернета, задействуют стандартный набор возможностей Windows и браузер Internet Explorer, полагая, что данные программы обеспечивают решение всех задач при работе с Web-ресурсами. Однако это не так. Существует целый

ряд полезных программ, дополняющих возможности браузера, – средства автономного просмотра, средства анализа обновления информации на конкретных узлах, переводчики и т.д.

Средства автономного просмотра Web-страниц

Одна из основных задач, выполняемых при работе в Web, – это просмотр и последующее копирование страниц на локальный диск пользователя. Используя обычный браузер, пользователю приходится скачивать страницу, затем копировать ее и переходить к следующей ссылке. При необходимости скопировать все документы большого сайта, данная работа становится отдельной проблемой. При копировании большого числа страниц требуется много времени и неизбежны пропуски и ошибки. Подобный подход приводит к неэффективному использованию сетевого времени. Поэтому, если вы пользуетесь каким-либо ресурсом, который редко обновляется, то намного выгоднее один раз скачать его на жесткий диск и просматривать локально, при этом скорость доступа к информации будет намного выше.

Иными словами, это удобно в том случае, если пользователю необходима информация, представленная относительно большими блоками.

Упомянутые выше программы называют *офлайновыми браузерами* или *Интернет-вампирами*.

При загрузке большого количества документов офлайновый браузер может быть эффективнее за счет возможности одновременной загрузки файлов с нескольких страниц сразу.

Функцию офлайнового браузера можно сравнить с библиотечной службой, которая предоставляет возможность скопировать часть библиотечного фонда и забрать его домой для работы в более оперативном режиме.

При копировании информации из Сети на жесткий диск могут возникать определенные проблемы с гиперссылками. Очевидно, что далеко не всегда удается скопировать все документы, на которые ссылаются страницы некоторого сайта. Так что отдельные ссылки останутся нерабочими. Более того, абсолютные ссылки указывают на определенные адреса URL. При копировании Web-страниц на локальный компьютер их адреса изменятся, и абсолютные ссылки перестанут работать в офлайновом режиме. Таким образом, для того чтобы сделать автономную копию, необходимо перенастраивать гиперссылки в копируемых документах. В этом случае в копии Web-узла на локальном компьютере может быть воссоздана структура взаимосвязи документов исходного сайта, однако копия для автономного просмотра уже не будет зеркальной копией Web-узла, с которого производилось копирование.

2.3.5. Служба имен доменов (DNS)

Человеку неудобно работать с числовым представлением *IP*-адреса, зато доменное имя запоминается легко, особенно если учесть, что, как правило, это имя содержательное.

С другой стороны, автоматическая работа серверов сети организована с использованием числового адреса. Благодаря ему промежуточные серверы могут осуществлять передачу запросов и ответов в нужном направлении, не зная, где именно находятся отправитель и получатель. Поэтому необходим перевод доменных имен в связанные с ними *IP*-адреса. Этим и занимаются серверы службы имен доменов *DNS* (*Domain Name System*). Запрос на получение одной из страниц сервера **www.abcde.com** (условный адрес) сначала обрабатывается сервером *DNS*, и далее он направляется по *IP*-адресу, а не по доменному имени.

2.3.6. Служба передачи файлов

Прием и передача файлов составляют значительный процент от прочих Интернет-услуг. Необходимость в передаче файлов возникает, например, при приеме файлов программ, при пересылке крупных документов, а также при передаче архивных файлов, в которых запакованы большие объемы информации.

Служба *FTP* имеет свои серверы в мировой сети, на которых хранятся архивы данных. Со стороны клиента для работы с серверами *FTP* может быть установлено специальное программное обеспечение, хотя в большинстве случаев браузеры *WWW* обладают встроенными возможностями, реализующими простейшие операции протокола *FTP*, например загрузку файлов с сервера.

Протокол *FTP* работает одновременно с двумя *TCP*-соединениями между сервером и клиентом. По одному соединению идет передача данных, а второе используется как управляющее. Протокол *FTP* также предоставляет серверу средства для идентификации обратившегося клиента. Этим часто пользуются коммерческие серверы и серверы ограниченного доступа, поставляющие информацию только зарегистрированным клиентам, – они выдают запрос на ввод имени пользователя и связанного с ним пароля. Однако существуют и десятки тысяч *FTP*-серверов с анонимным доступом для всех желающих. В этом случае в качестве имени пользователя надо ввести слово «anonymous», а в качестве пароля задать адрес электронной почты. В большинстве случаев программы-клиенты *FTP* делают это автоматически.

2.3.7. ICQ

Название службы является акронимом выражения *I seek you – я тебя ищу*. Эта служба позволяет пользователями, присутствующим в данный момент в Интернете, общаться в реальном времени.

Необходимость в подобной услуге связана с тем, что большинство пользователей не имеют постоянного *IP*-адреса. Для пользования этой службой надо зарегистрироваться на ее центральном сервере (*http://www.icq.com*) и получить персональный идентификационный номер *UIN* (*Universal Internet Number*). При каждом подключении к Интернету программа *ICQ*, установленная на компьютере, определяет текущий *IP*-адрес и сообщает его центральной службе, которая, в свою очередь, оповещает партнеров по контактам. Далее партнеры (если они тоже являются клиентами данной службы) могут установить с нужным пользователем прямую связь. Программа предоставляет возможность выбора режима связи («готов к контакту»; «прошу не беспокоить, но готов принять срочное сообщение»; «закрыт для контакта» и т.п.). После установления контакта происходит прямое общение в режиме реального времени.

Кроме того, номер *UIN* можно сообщить партнерам по контактам, и тогда служба *ICQ* приобретает характер Интернет-пейджера. Зная номер *UIN* партнера, но не зная его текущий *IP*-адрес, можно через центральный сервер службы отправить ему сообщение с предложением установить соединение.

2.3.8. Списки рассылки и сетевые новости

Списки рассылки. Списки рассылки позволяют, послав письмо на один адрес, в действительности разослать его целой группе получателей. Служба предоставляет возможность подписаться на тот или иной список рассылки и, соответственно, присоединиться к проводимым в электронной почте дискуссиям на различные темы.

Схема работы сервиса показана на рисунке II.2.12. Пользователь при помощи почтового клиента направляет письмо на сервер, в базе которого хранится список рассылки (*Mailing list*), после чего письмо автоматически направляется всем участникам. Существуют списки рассылки по различным темам. Подписчиками обычно являются специалисты, интересующиеся новостями в данной области. Списки бывают двух типов: вещательного (дают возможность только получать информацию) и дискуссионного (можно также посылать сообщения всем участникам списка).

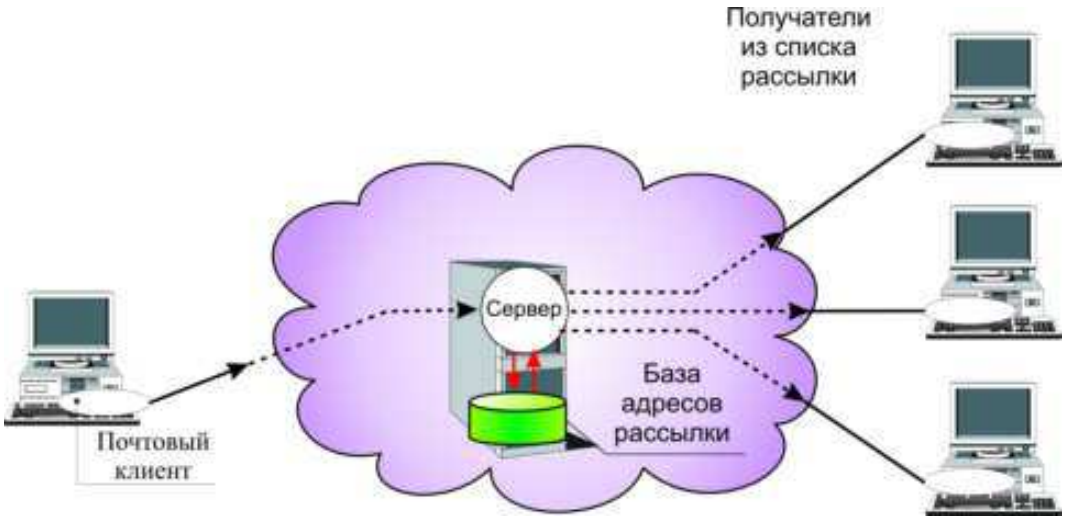


Рис. II.2.12. Схема работы сервиса списков рассылки

Некоторые списки рассылки администрируются *модератором* – человеком, который читает все сообщения перед тем, как они отправляются участникам, и имеет право отклонить сообщения, не подходящие для данного списка.

В настоящее время существуют тысячи списков рассылки, на которые можно свободно подписаться. Большинство людей являются абонентами того или иного списка рассылки, иногда даже не задумываясь об этом. Подтверждая свое желание получать письма по некоторой тематике, пользователи автоматически попадают в некий список рассылки и становятся абонентами коллективной почтовой рассылки. Информация отправляется каждому лицу из данного списка, хранящегося на сервере. Многие компании стремятся создать базу адресов своих клиентов и информировать их о новых сервисах и услугах посредством списков рассылки.

Сетевые новости.

Сетевые новости (*Netnews*) или группы новостей (*Newsgroups*) – это глобальная система *Интернет-конференций*, которая позволяет организовать текстовые дискуссии в рамках тематических групп. Информация, которую можно прочитать в группе новостей, – это необязательно новости в привычном смысле слова. Это может быть любое сообщение, поданное для обсуждения в данную группу новостей.

Сервис *Newsgroups* имеет сходные черты со службой списков рассылки, но если сообщения, распространяемые по списку рассылки, приходят в почтовый ящик абонента и могут храниться на сервере сколь угодно долго, то статья, посылаемая в *Newsgroups*, становится доступной для всех участников группы новостей на определенный срок. Если списки рассылки

расположены на одном сервере, который принадлежит конкретной организации, то *Netnews* не хранятся на одном сервере, а копируются на тысячи серверов по всему миру. Поэтому адресаты, которые прочитают то или иное сообщение, заранее не известны.

Netnews отличается от почтовых списков рассылки именно неопределенностью круга участников и отсутствием центрального сервера, хранящего список подписчиков. Каждый может подключиться к группе новостей и присоединиться к обсуждению темы, в которой принимает участие множество людей. Для работы с сетевыми новостями необходима специальная клиентская программа.

Клиентская программа для работы с группами новостей называется *Newsreader*. Она позволяет читать сообщения, удалять их, отвечать на статьи и посылать новые в группы новостей.

В ряде случаев одна программа может интегрировать сразу несколько сервисов. Например, Outlook Express, помимо почтового клиента, содержит программу для чтения новостей.

Существует огромное количество новостных серверов, которые образуют целую инфраструктуру, обеспечивающую механизм передачи новостей по всему миру.

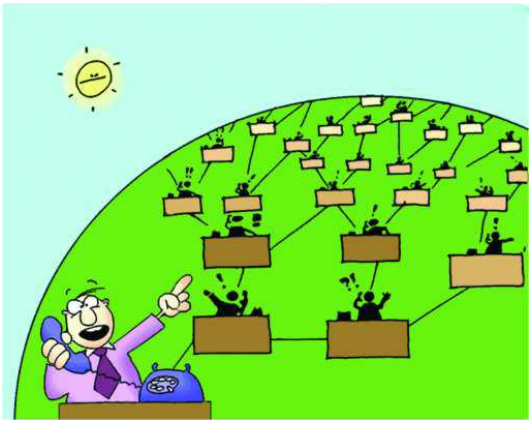


Рис. II.2.13. Процесс передачи статей в Newsgroup

Процесс передачи статей в *Newsgroups* можно сравнить с распространением слухов: каждый узел сети, узнавший что-то новое (т.е. получивший сообщение), передает новость всем узлам, с которыми он обменивается новостями (рис. II.2.13).

Посланная клиентом статья вначале попадает в локальную базу сервера, на который приходит сообщение. После этого сервер, получивший статью, передает ее соседям («подписанным» на одну из групп, к которым принадлежит статья), те записывают статью в свою базу и передают ее своим соседям. Процесс продолжается до тех пор, пока статья не окажется на всех серверах, участвующих в передаче.

Ответы на статью могут быть представлены в двух вариантах: в виде отзыва в *Newsgroups* либо в виде электронного письма, отправленного непосредственно автору статьи. Хорошим тоном считается ситуация, когда отвечающие пишут на личный адрес электронной почты задавшего вопрос

(это позволяет избежать массы идентичных ответов в группу), а человек, получивший ответы, отправляет обобщенный результат в группу новостей, чтобы все участники конференции могли с ним ознакомиться (рис. II.2.14).

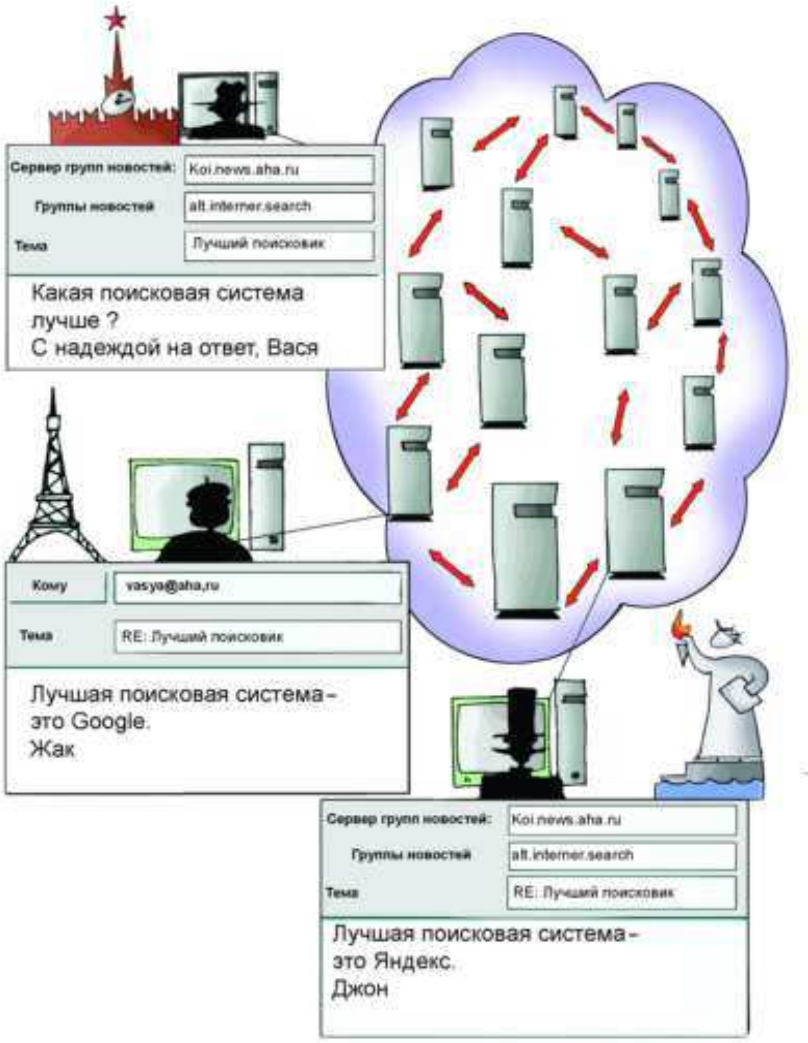


Рис. 2.14. Принцип функционирования *Newsgroups*

2.4. Поиск информации в Интернете

2.4.1. Индексированные каталоги

Каталог представляет собой данные, структурированные по темам в виде иерархических структур. Тематические разделы первого уровня определяют наиболее популярные, максимально широкие темы, такие как «спорт», «отдых», «наука», «магазины» и т.д. В каждом таком разделе есть подразделы. Таким образом, пользователь может уточнять интересующую его область, путешествуя по дереву каталога и постепенно сужая область поиска. Дойдя до нужного подкаталога, пользователь находит в нем набор ссылок.

Пример. 2.1. При поиске информации о ноутбуках цепочка поиска может выглядеть следующим образом: *Информационные технологии → Компьютеры → Ноутбуки.*

Обычно в каталоге все ссылки являются профильными, поскольку составлением каталогов занимаются не программы, а люди. Очевидно, что если ведется поиск общей информации по некоторой широкой теме, то целесообразно обратиться к каталогу. Если же необходимо найти конкретный документ, то каталог окажется малоэффективным поисковым средством.

Существует огромное количество каталогов. Один из наиболее популярных каталогов находится по адресу http://list.mail.ru/.

Помимо каталогов общего профиля, в Сети достаточно много специализированных каталогов. Например, по адресу http://www.kinder.ru/ можно найти прекрасный каталог, посвященный детским ресурсам. В случае если внутри отдельной темы каталога находится огромное количество ресурсов, возникает проблема выбора. В некоторых каталогах имеется сортировка по популярности, например, в каталоге Яндекса сортировка идет по индексу цитирования – числу ссылок на сайт с других сайтов.

2.4.2. Поисковые машины.
Принцип работы поисковой машины.
Язык запросов

Если бы компьютер был высокоинтеллектуальной системой, которой можно было легко объяснить, что ищет пользователь, то он выдавал бы два-три документа – именно те, которые нужны. Но, к сожалению, это не так, и в ответ на запрос пользователь обычно получает длинный список документов, многие из которых не имеют никакого отношения к тому, о чем он спрашивал. Такие документы называются *нерелевантными* (от англ. *relevant* – подходящий, относящийся к делу).

Таким образом, *релевантный документ* – это документ, содержащий искомую информацию. Очевидно, что от умения грамотно формулировать запрос зависит процент получаемых релевантных документов. Доля релевантных документов в списке всех найденных поисковой машиной документов называется *точностью поиска*. Нерелевантные документы называют *шумовыми*. Если все найденные документы релевантные (шумовые отсутствуют), то *точность поиска составляет 100%*. Если найдены все релевантные документы, то *полнота поиска – 100%*.

Таким образом, качество поиска определяется двумя взаимозависимыми параметрами: точностью и полнотой поиска. Увеличение полноты поиска снижает точность, и наоборот.

Как работает поисковая машина. Поисковые системы можно сравнить со справочной службой, агенты которой обходят предприятия, собирая информацию в базу данных (рис. II.2.15). При обращении в службу информация выдается из этой базы. Данные в базе устаревают, поэтому агенты их периодически обновляют. Некоторые предприятия сами присылают данные о себе, и к ним агентам приезжать не приходится. Иными словами, справочная служба имеет две функции:

1) создание и постоянное обновление данных в базе;
2) поиск информации в базе по запросу клиента.

Рис. II.2.15. Поисковые системы как справочная служба

Аналогично, поисковая машина состоит из двух частей: так называемого робота (или паука) и поискового механизма. Робот обходит серверы Сети и формирует базу данных поискового механизма.

База робота в основном формируется им самим (робот сам находит ссылки на новые ресурсы) и в гораздо меньшей степени – владельцами ресурсов, которые регистрируют свои сайты в поисковой машине. Помимо робота (сетевого агента, паука, червяка), формирующего базу данных, существует программа, определяющая рейтинг найденных ссылок.

Принцип работы поисковой машины сводится к тому, что она опрашивает свой внутренний каталог (базу данных) по ключевым словам, которые пользователь указывает в поле запроса, и выдает список ссылок, ранжированный по релевантности.

Следует отметить, что, отрабатывая конкретный запрос пользователя, поисковая система оперирует именно внутренними ресурсами (а не

пускается в путешествие по Сети, как часто полагают неискушенные пользователи), а внутренние ресурсы, естественно, ограничены. Несмотря на то, что база данных поисковой машины постоянно обновляется, поисковая машина не может проиндексировать все Web-документы: их число слишком велико. Поэтому всегда существует вероятность, что искомый ресурс просто неизвестен конкретной поисковой системе.

Эту мысль наглядно иллюстрирует рисунок II.2.16. Эллипс 1 ограничивает множество всех Web-документов, существующих на некоторый момент времени; эллипс 2 – все документы, которые проиндексированы данной поисковой машиной; а эллипс 3 – искомые документы. Таким образом, найти с помощью данной поисковой машины можно лишь ту часть искомых документов, которые ею проиндексированы.

Рис. II.2.16. Схема, поясняющая возможности поиска

Проблема недостаточности полноты поиска состоит не только в ограниченности внутренних ресурсов поисковика, но и в том, что скорость робота ограниченна, а количество новых Web-документов постоянно растет. Увеличение внутренних ресурсов поисковой машины не может полностью решить проблему, поскольку скорость обхода ресурсов роботом конечна.

Неправильно было бы считать, что поисковая машина содержит копию исходных ресурсов Интернета. Полная информация (исходные документы) хранится отнюдь не всегда, чаще хранится лишь ее часть – так называемый индексированный список, или индекс, который гораздо компактнее текста документов и позволяет быстрее отвечать на поисковые запросы.

Для построения индекса исходные данные преобразуются так, чтобы объем базы был минимальным, а поиск осуществлялся очень быстро и давал максимум полезной информации. Объясняя, что такое индексированный список, можно провести аналогию с его бумажным аналогом – так называемым *конкордансом*, т.е. словарем, в котором в алфавитном порядке перечислены слова, употребляемые конкретным писателем, а также указаны ссылки на них и частота их употребления в его произведениях.

Очевидно, что конкорданс (словарь) гораздо компактнее исходных текстов произведений и найти в нем нужное слово намного проще, нежели перелистывать книгу в надежде наткнуться на нужное.

Построение индекса. Схема построения индекса показана на рисунке II.2.17. Сетевые агенты, или *роботы-пауки*, «ползают» по Сети, анализируют содержимое Web-страниц и собирают информацию о том, что и на какой странице было обнаружено.

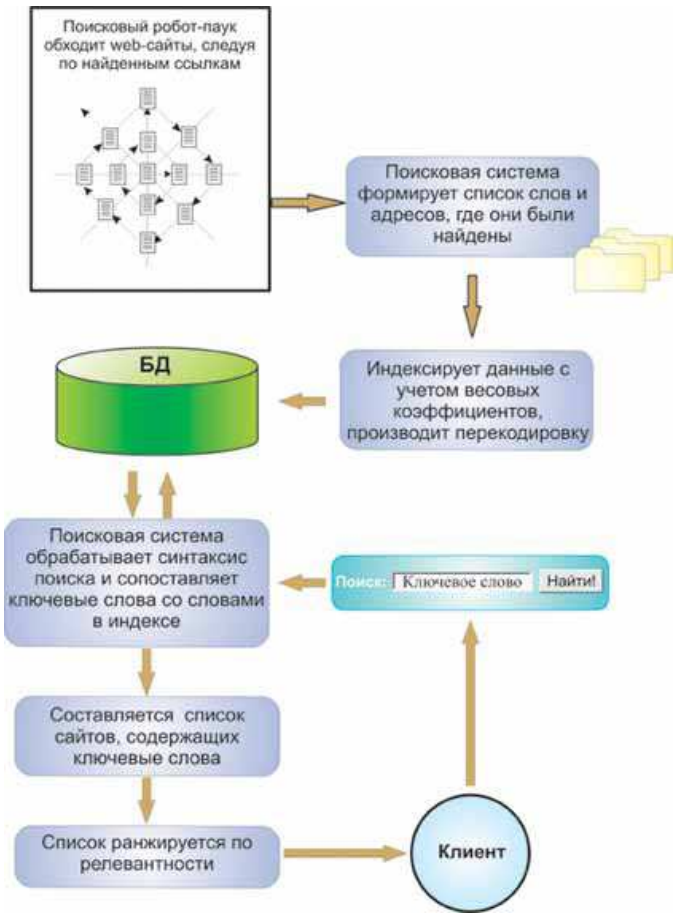


Рис. II.2.17. Создание базы поиска роботами-пауками

При нахождении очередной HTML-страницы большинство поисковых систем фиксируют слова, картинки, ссылки и другие элементы (в раз-

ных поисковых системах по-разному), содержащиеся на ней. Причем при отслеживании слов на странице фиксируется не только их наличие, но и местоположение, т.е., где эти слова находятся:

– в заголовке (title),
– подзаголовках (subtitles),
– в метатэгах (meta tags) или в других местах.

При этом обычно фиксируются значимые слова, а союзы и междометия типа «а», «но» и «или» игнорируются. Метатэги позволяют владельцам страниц определить ключевые слова и тематику, по которым индексируется страница. Это может быть актуально в случае, когда ключевые слова имеют несколько значений. Метатэги могут сориентировать поисковую систему при выборе из нескольких значений слова на единственно правильное. Однако они работают надежно только в том случае, когда заполняются честными владельцами сайта. Недобросовестные владельцы Web-сайтов помещают в свои метатэги наиболее популярные в Сети слова, не имеющие ничего общего с темой сайта. В результате посетители попадают на незапрашиваемые сайты, повышая тем самым их рейтинг. Именно поэтому многие современные поисковики либо игнорируют метатэги, либо считают их дополнительными по отношению к тексту страницы.

Очевидно, что если вы ищете сайты по ключевому слову «собака», то поисковый механизм должен найти не просто все страницы, где упоминается слово «собака», а те, где это слово имеет отношение к теме сайта. Для того чтобы определить, в какой степени то или иное слово имеет отношение к профилю некоторой Web-страницы, необходимо оценить, насколько часто оно встречается на странице, есть ли по данному слову ссылки на другие страницы или нет – необходимо ранжировать найденные на странице слова по степени важности. Словам присваиваются весовые коэффициенты в зависимости от того, сколько раз и где они встречаются (в заголовке страницы, в начале или в конце страницы, в ссылке, в метатэге и т.п.). Каждый поисковый механизм имеет свой алгоритм присваивания весовых коэффициентов – это одна из причин, по которой поисковые машины по одному и тому же ключевому слову выдают различные списки ресурсов.

Поскольку страницы постоянно обновляются, процесс индексирования должен выполняться постоянно. Роботы-пауки путешествуют по ссылкам и формируют файл, содержащий индекс, который может быть доволь-

но большим. Для уменьшения его размеров прибегают к минимизации объема информации и сжатию файла. Имея несколько роботов, поисковая система может обрабатывать сотни страниц в секунду. Сегодня мощные поисковые машины хранят сотни миллионов страниц и получают десятки миллионов запросов ежедневно.

При построении индекса решается также задача снижения количества дубликатов – задача нетривиальная, учитывая, что для корректного сравнения нужно сначала определить кодировку документа. Еще более сложной задачей является отделение очень похожих документов (их называют «почти дубликаты»), например таких, в которых отличается лишь заголовок, а текст дублируется. Подобных документов в Сети очень много – например, кто-то списал реферат и опубликовал его на сайте за своей подписью. Современные поисковые системы позволяют решать подобные проблемы.

Поиск по индексу. Поиск по индексу заключается в том, что пользователь формирует запрос и передает его поисковой машине. В случае когда у пользователя имеется несколько ключевых слов, весьма полезно использование булевых операторов.

Наиболее часто используемые булевы операторы:

– **AND** – все термины, соединенные **AND**, должны присутствовать в предлагаемом документе. Большинство поисковых систем используют значок «+» вместо **AND**;

– **OR** – как минимум одно из ключевых слов, соединенных **OR**, должно присутствовать в искомом документе;

– **NOT** – ключевое слово (слова), следующее за **NOT**, не должно появляться в искомом документе. Некоторые поисковые системы используют значок «–» вместо «NOT»;

– **FOLLOWED BY** – одно из ключевых слов должно следовать непосредственно за другим;

– **NEAR** – одно из слов должно отстоять на определенное количество слов от другого;

– **кавычки** – слова внутри кавычек – это фраза, которая целиком должна быть найдена в пределах документа или файла.

Текст, в пределах которого проверяется действие логических операторов, называется единицей поиска. Это может быть предложение, абзац или весь документ. В разных поисковых системах могут использоваться различные единицы поиска.

Пример. 2.2. Можно искать документы, в которых два слова – «электрический» и «счетчик» – находятся одновременно в пределах предложения или в пределах всего документа. Соответственно поиск в пределах предложения возможен для тех систем, которые имеют в индексе подробный адрес.

Синтаксис языка запросов в разных поисковых системах может отличаться, обычно в справочных данных на поисковом сервере приводится информация о синтаксисе запросов.

2.4.3. Онлайновые справочники и энциклопедии

В ряде случаев бывает нужно найти не просто документ, содержащий ключевое слово, а именно толкование некоторого слова. При поиске незнакомого термина с помощью поисковой машины можно получить целый ряд статей, в которых этот термин используется, и при этом так и не узнать, что же он все-таки обозначает.

В ряде случаев можно воспользоваться поиском с ключевыми словами типа «что такое (неизвестный термин)», «(неизвестный термин) – это» или «(неизвестный термин) представляет собой» и т.д.

Однако, если это не новый термин, предпочтительнее начать подобный поиск в онлайновой энциклопедии.

Одной из крупнейших онлайновых энциклопедий является ресурс «Яндекс. Энциклопедии» (http://encycl.yandex.ru/). Этот проект содержит 14 энциклопедий, в том числе статьи из Большой Советской Энциклопедии и «Энциклопедию Брокгауза и Эфрона». К крупным относится и «Энциклопедия Кирилла и Мефодия», которую можно найти по адресу http://www.km.ru/.

Особенно актуальным является поиск толкований терминов по информационным технологиям, которые развиваются так быстро, что уследить за их появлением очень сложно. К сожалению, большинство словарей из данной категории – англоязычные. Весьма популярным и объемным является англоязычный *FOLDOC* (*Free On-line Dictionary Of Computing* – http://wombat.doc.ic.ac.uk/foldoc/index.html) – более 13 тысяч терминов. Следует рекомендовать еще как минимум два онлайновых словаря: *Webopedia* и *WhatIs.com.*

Помимо традиционного словаря, ресурс Webopedia (http://www.pcwebopaedia.com/) имеет массу специализированных сервисов, например: «Кто есть кто в компьютерных технологиях», «Сравнительная таблица микропроцессоров», «История развития компьютерных технологий» и др.

3. Технология создания Web-страниц

3.1. Введение в HTML

Итак, что же такое HTML? Аббревиатура *HTML (HyperText Markup Language)* как раз и означает язык гипертекстовой разметки. Создал его всем известный основоположник «всемирной паутины» *Тим Бернерс-Ли* (на основе уже имеющегося к тому времени языка SGML), который и сейчас продолжает участвовать в работе над новыми стандартами языка гипертекстовой разметки в рамках консорциума W3C.

Собственно говоря, первая версия языка HTML появилась в начале девяностых годов прошлого века и была ориентирована в первую очередь на передачу информации в научной среде.

До этого Интернет был уделом немногих знающих и интересующихся людей, но с появлением языка HTML и первых браузеров, способных интерпретировать его код в понятные и удобные пользователю вещи (*web-страницу* или как еще часто говорят – *web-документ*), всемирная паутина получила мировое распространение.

Довольно интересными представляются темп и нюансы развития этого языка гипертекстовой разметки. Итак, через несколько лет после появления первой версии HTML, по инициативе Тима Бернерса-Ли был создан консорциум *W3C* (*World Wide Web Consortium* – консорциум всемирной паутины), который призван был стать законодателем стандартов языка гипертекстовой разметки.

В 1994 году разрабатываются стандарты языка гипертекстовой разметки второй версии, а уже в 1995 ведутся работы над HTML 3 с поддержкой CSS (таблиц каскадных стилей). Примерно в это же время появляется и набирает популярность первый браузер *Мозаика*, который очень быстро был преобразован в *Netscape Navigator*. Microsoft хотела купить Netscape Navigator для интеграции его в Windows, но разработчики этого браузера отказались, в результате чего мы получили собственное творение Microsoft (Internet Explorer – IE), которое они создали на базе открытых кодов *Мозаики*.

Однако IE (в силу своей предустановленности в самой популярной операционной системе) сумел вытеснить с рынка браузеров некогда очень популярный Netscape Navigator, но получил взамен ряд новых конкурентов (Opera, Mozilla и др.). В этот период разработчики браузеров зачастую опережали W3C и вводили свои собственные стандарты, ибо работа над форматом HTML в W3C шла довольно медленно.

В ответ W3C в течение одного 1997 года сделал огромный рывок – язык гипертекстовой разметки претерпел сразу два изменения, перейдя от версии 3.2 до 4.0, а затем (в 1999) и до версии, которую мы используем до настоящего времени, – HTML 4.01.

Сейчас W3C с подачи конгломерата разработчиков браузеров ведет активные работы над форматом HTML 5, но ждать его появления в ближайшем будущем, наверное, не стоит. Хотя некоторые нововведения формата HTML 5 уже поддерживаются в той или иной степени некоторыми браузерами. Скорее всего, именно так и будет происходить дальше – HTML 5 будет внедряться по частям, но полной его поддержки всеми браузерами придется ждать довольно долго.

Итак, с помощью языка гипертекстовой разметки мы создаем web-страницы (web-документы). Браузер получает *html-код* web-документа, разбирает этот код, подгружает все необходимые дополнительные элементы оформления web-страницы (изображения, css-файлы, скрипты), и пользователь видит на экране страницу сайта.

3.2. Структура Web-документа. Назначение основных тэгов HTML

Для того чтобы пояснить идею организации HTML-документа, уместно привести аналогию с работой редактора издательства, который готовит текст для верстки. Редактор имеет исходный текст и делает его разметку, в которой отмечает, как необходимо отформатировать те или иные элементы и как изменить порядок следования абзацев. Например, это могут быть пометки типа: «данный абзац напечатать жирным шрифтом, такое-то предложение выделить красным цветом, а после абзаца такого-то вставить текст со страницы такой-то». Верстальщик читает комментарии редактора и отображает финальный вариант документа в том виде, в котором он потом выйдет из печати.

Аналогичные комментарии записываются в HTML-документе, затем они прочитываются браузером, для того чтобы браузер «знал», как нужно отображать данные на экране компьютера.

Можно сказать, что браузер выполняет форматирование документа исходя из набора инструкций, содержащихся в HTML, и из возможностей отображения информации конкретным компьютером. Например, если на компьютере нет шрифта, которым предписывается отобразить тот или иной текст, будет использован другой подходящий шрифт из доступного набора.

Кроме того, пользователи имеют возможность изменить настройки браузера так, чтобы документ отображался оптимальным образом. Например, в настройках браузера можно указать правила выбора цвета для отображения фона документа или указать, каким цветом должны выделяться текстовые гиперссылки. Очевидно, что один и тот же HTML-документ будет отражаться браузерами с разной настройкой по-разному.

На рисунке II.3.1 приведена наглядная структура HTML-документа.

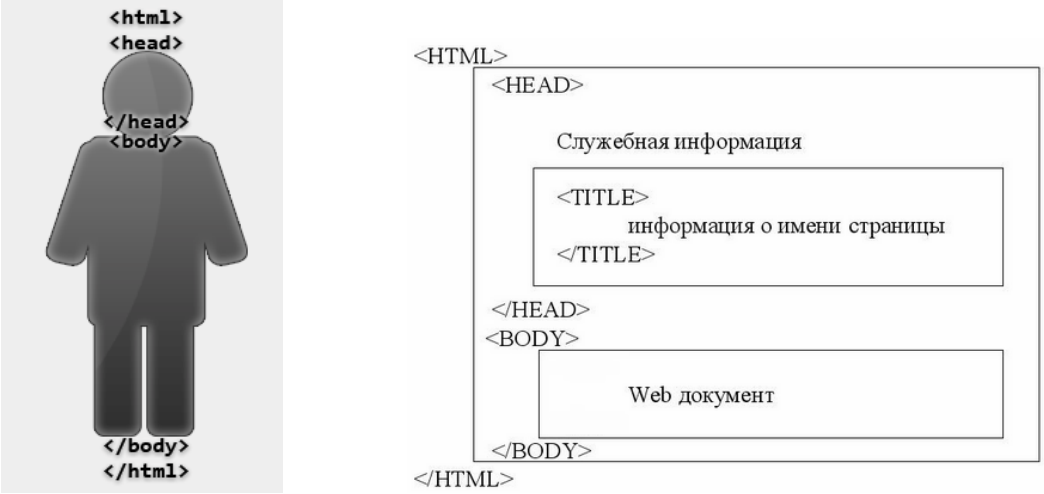


Рис. II.3.1. Наглядная структура HTML-документа

Вот как может выглядеть код самой простой html-страницы (рис. II.3.2):

```
<html>
<head>
<title>Мой первый сайт</title>
</head>
<body>
Привет всем! А это мой первый сайт
</body>
</html>
```

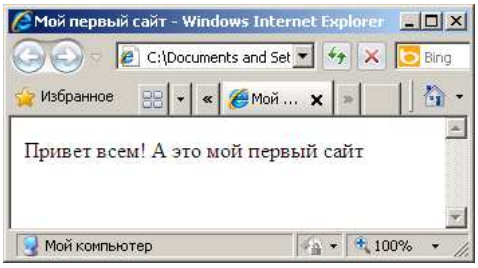


Рис. II.3.2. Вид страницы «Мой первый сайт»
в окне браузера

Рассмотрим подробно структуру приведенной веб-страницы:

– тэг **<html>** сообщает о том, что начинается структура html-документа, **</html>** – что она заканчивается;

– между тэгами **<head> </head>** хранится, по большей части, информация для браузера и поисковых систем;

– в тэгах **<title> </title>** содержится заголовок нашей страницы;

– тэг **<body>** говорит о том, что далее идет информация предназначенная для пользователя, **</body>** говорит о том, что информация для пользователя заканчивается.

Тэг (от англ. *tag* – ярлык, маркер) – элемент языка разметки гипертекста. Для того чтобы тэги отличались от обычного текста, их заключают в угловые скобки (< и >**)**.

Существует более сотни тэгов для разметки информации на Web-странице. Все тэги делятся на два типа:

– «одиночные»;

– «закрывающиеся».

Большинство тэгов используются парами, т.е. являются «закрывающимися». Открывающий и закрывающий тэги начинаются и завершаются угловыми скобками. Закрывающий тэг отличается от открывающего наличием символа «/» (слэш) перед ключевым словом. Например, пара тэгов **…** используется для того, чтобы дать браузеру команду отобразить текст, помещенный между тэгами, жирным шрифтом, а тэги **…** показывают, что при отображении текста должен быть применен курсив. Открывающий и закрывающий тэги воздействуют на часть документа, заключенную между ними.

Одним из важнейших является тэг, определяющий гиперссылки. При определении гиперссылки необходимо связать элемент Web-страницы, по которому будет происходить переход, с адресом данного перехода. Для этого используют пару, которая начинается с открывающего тэга **<a>**, по первой букве слова *anchored* (от англ. «привязанный»), и заканчивается закрывающим тэгом ****.

Адрес перехода определяется значением атрибута **href** и представляет собой URL того ресурса, на который указывает ссылка. Таким образом, в простейшем случае определение гиперссылки выглядит как:

имя ссылки

Более подробно работа с HTML будет рассмотрена в лабораторном практикуме.

3.3. Общая характеристика программ для создания Web-документов

Популярность службы WWW во многом определяется тем, что пользователи могут не только просматривать сайты, но и создавать собственные. Процесс создания и функционирования Web-сайта иллюстрируется схемой, изображенной на рисунке II.3.3.

У каждого сайта есть разработчик, который на своем компьютере готовит исходные материалы: графику, медиаданные (звук, анимация), используя соответствующие редакторы, и непосредственно разрабатывает сайт, применяя тот или иной HTML-редактор. Обычно сайт разрабатывается на локальном компьютере, тестируется, а затем переносится на Web-сервер при помощи FTP-протокола. После опубликования материал становится доступен для всех посетителей. Если пользователи имеют право только на просмотр сайта, то разработчики имеют возможность обновлять и изменять его содержимое.

Рис. II.3.3. Схема разработки и просмотра Web-сайта

Следует отметить, что создать HTML-документ можно с использованием обычного текстового редактора. Например, с помощью входящей в Windows программы *Notepad*. В данном случае разработчик пишет HTML-код «с нуля», не используя ни шаблонов, ни подсказок.

Особо следует отметить офисные приложения: все приложения Microsoft Office позволяют сохранить результат работы в HTML-формате. Многие домашние пользователи выбирают Microsoft Word в качестве средства разработки простых Web-страниц. Правда HTML-код в этом случае не будет оптимальным.

Когда пользователь сохраняет документы в HTML-формате с помощью Microsoft Word, в него добавляются дополнительные тэги, для того чтобы впоследствии можно было использовать всю функциональность Word для дальнейшей работы с содержимым документа.

Для того чтобы уменьшить размер Web-странички, в Microsoft Word 2003 можно сохранять данные в специальном фильтрованном формате (*filtered HTML*), в этом случае лишние тэги будут удалены.

Правда, если открыть Web-страницу, сохраненную в режиме *filtered HTML* в программе Word, вероятно, некоторые возможности редактирования этого документа будут утрачены. Поэтому следует использовать режим *filtered HTML* только на финальной стадии, перед непосредственным переводом в HTML.

Несмотря на то, что простые Web-странички можно создавать, не прибегая к помощи специализированных HTML-редакторов, заниматься разработкой и поддержкой более-менее серьезных сайтов без них невозможно.

Все имеющиеся на рынке HTML-редакторы можно разделить на две большие группы:

– программы, имеющие в своем составе визуальные редакторы (*design-based editors*) – средства, которые автоматически формируют необходимый HTML-код, позволяя разрабатывать Web-страницы в режиме *WYSIWYG* (*What You See Is What You Get – что вижу, то и получаю*);

– программы-редакторы, которые предоставляют редактор и вспомогательные средства для автоматизации написания кода (*code-based editors*).

Возникает вопрос, какой подход лучше – визуальный или основанный на написании кода? У каждого из методов есть сторонники и противники.

Сторонники *визуального редактирования* предпочитают данный подход в силу простоты, наглядности и доступности для широкой аудитории начинающих пользователей.

Сторонники *кодирования* убеждены, что только вручную можно создать оптимальный код и что программная разметка точнее визуальной. Те, кто предпочитает создание сайта вручную, критикуют редакторы, предоставляющие готовые шаблоны, формы и заготовки, за однотипность.

Необходимо учитывать, что сегодня многие пользователи (разного уровня подготовки) занялись созданием собственных сайтов и нуждаются в различных по степени сложности продуктах. Одним пользователям требуется продукт «на вырост», другие хотят создать двухстраничный персональный сайт и больше к этому не возвращаться.

Отметим, наконец, что существуют и компромиссные решения: например, приложения *Microsoft FrontPage* и *Macromedia Dreamweaver* имеют не только визуальные средства, но и весьма развитые редакторы кода. В последней версии FrontPage 2003 реализован даже сдвоенный режим, в котором пользователь может наблюдать, как действия визуального редактора приводят к изменению кода.

При этом очевидно, что чем выше профессионализм пользователя, тем меньше готовых шаблонов, макетов и подсказок ему требуется.

Таким образом, простой текстовый редактор – это незаменимое средство для Web-дизайнера, досконально знающего HTML, но не слишком полезное для человека, который не собирается его изучать. Рассмотрим наиболее популярные редакторы.

3.3.1. FrontPage

Разработчик: Microsoft.

Microsoft FrontPage – это простой в освоении и удобный Web-редактор для проектирования, подготовки и публикации Web-сайтов. Благодаря интеграции с семейством продуктов MS Office, привычному интерфейсу и обилию шаблонов программа позволяет быстро освоить работу даже начинающим пользователям, если они знакомы с основами работы в MS Word. При этом FrontPage нельзя назвать «решением для начинающих»: программа предоставляет широкие функциональные возможности и разнообразные средства оптимизации коллективной разработки, а также позволяет быстро создавать динамические комплексные Web-узлы практически любой сложности.

Усовершенствованная поддержка графики упрощает работу во FrontPage 2003 с изображениями из других приложений. FrontPage 2003 удачно сочетает в себе возможности использования визуального конструктора и средства редактирования кода. В этой версии программы появилось

разделенное окно (режим *Split*), которое состоит из двух областей – «Конструктор» («Design») и «Код» («Code»). В окне «Design» ведется разработка в режиме WYSIWYG, а в окне «Code» идет автоматическое обновление кода при внесении изменений в макет.

Комплексные средства проектирования позволяют повысить качество создаваемого кода и усовершенствовать навыки программирования.

FrontPage 2003 генерирует эффективный HTML-код, не содержащий избыточности, характерной для кода, генерируемого Microsoft Word. Средства написания сценариев обеспечивают возможность интерактивного общения с посетителями.

Функция интеллектуального поиска и замены осуществляет поиск и замену атрибутов или тэгов на заданных страницах. При этом можно указывать сложные правила поиска и замены, что позволяет быстро выполнять обновления Web-узла. FrontPage предоставляет возможность оптимизации HTML-кода, написанного в других приложениях за счет удаления избыточных тэгов, пробелов и т.п.

3.3.2. Macromedia Dreamweaver MX

Разработчик: Macromedia.

Dreamweaver всегда считался инструментом скорее дизайнера, чем программиста. Однако в последней версии имеется полный набор средств как для визуального конструирования, так и для кодирования. Dreamweaver MX 2004 v7.0 – это приложение, в котором разработчик может работать в одной среде, быстро создавая, развивая и обслуживая Web-сайт и Интернет-приложения с помощью средств визуального редактирования, разработки приложений и быстрого написания кода, реализованных в едином интегрированном решении. Разработчики могут использовать Dreamweaver с различными серверными технологиями и создавать мощные Интернет-приложения, которые обеспечивают пользователям доступ к базам данных и Web-сервисам.

Одним из важных нововведений в Dreamweaver MX 2004 v7.0 является *кросс-браузерный валидатор* (*cross-browser validator*) – средство, позволяющее автоматически проверять корректность тэгов на совместимость для различных Web-браузеров.

3.3.3. HomeSite

Разработчик: Macromedia.

В то время как Dreamweaver – это, прежде всего, визуальный редактор, программа HomeSite предназначена для разработчиков, которые пред-

почитают работать с исходными кодами и обычно используют не только HTML. Macromedia удалось усовершенствовать данный редактор для работы с большими сайтами.

HomeSite – это HTML-редактор для профессиональных разработчиков. Web-разработчики, которые пишут код HTML, высоко оценили HomeSite. Этот кодовый редактор, имеющий массу настроек, прекрасно работает с другими приложениями Macromedia.

Еще одним плюсом HomeSite является понятный, удобный интерфейс, который позволяет редактировать несколько документов одновременно. Программа имеет встроенные средства проверки кода, предоставляет подсказки тэгов, обеспечивает проверку и оптимизацию кода.

3.3.4. CoffeeCup HTML Editor

Разработчик: CoffeeCup Software.

CoffeeCup не имеет визуальных средств редактирования, но, как редактор кода, снабжена очень удачным набором функций. Программа предлагает многочисленные графические заготовки, встроенные Web-шаблоны, средства опубликования материалов на сайте, встроенный оптимизатор HTML-кода, настраиваемый под пользователя интерфейс и многое другое.

3.4. Качество представления текстов на Web-сайте

Как показывают исследования, пользователи читают Web-страницы более бегло, чем печатные материалы. Лишь 16% пользователей при просмотре Web-страницы читают каждое слово.

Поскольку Web-информация по своей природе интерактивна, у человека пропадает привычка читать текст от начала до конца. Вместо того чтобы прочитывать страницы до конца, человек пытается ухватить на каждой что-то наиболее ценное.

Длинный неформатированный текст – это та часть web-страницы, на которую не обратят внимания.

Таким образом, Web-страницы необходимо писать так, чтобы они *легко усваивались при беглом просмотре*. Для этого следует использовать:

- обозримый объем;
- выделение ключевых слов;
- выразительные подзаголовки;
- списки с маркерами;
- один абзац на одну идею.

Грамотное форматирование вызывает у читателя доверие к информации. Это особенно важно, когда изначально неизвестно, кто стоит за конкретным ресурсом.

3.5. Измерение эффективности (юзабилити) стиля Web-страниц

Для того чтобы продемонстрировать зависимость степени усвоения материала, представленного на Web-страницах, от качества представления информации, уместно привести пример исследования Якоба Нильсена и Джона Моркеса (табл. II.3.1).

Таблица II.3.1.

Результаты измерений юзабилити

Стиль сайта	Абзац для примера	Улучшение юзабилити (относительно контрольного примера)
1	2	3
Рекламный стиль. Контрольный пример с большим количеством «рекламной воды», который можно встретить на многих коммерческих сайтах	В штате Небраска расположены знаменитые на весь мир ландшафты, которые неизменно притягивают к себе толпы людей. В 1996 году одним из самых популярных мест был Парк форта Робинсона (Fort Robinson State Park) (355 000 посетителей), Исторический музей и парк Arbor Lodge (Arbor Lodge State Historical Park & Museum) (100 000), Carhenge (86 598), Музей пионеров прерии (Stuhr Museum of the Prairie Pioneer) (60 000) и исторический парк Ранчо Буффало Билла(Buffalo Bill Ranch State Historical Park) (28 446)	0% (по определению)
Сжатый текст. Количество слов по сравнению с исходным примером сокращено наполовину	В 1996 году самыми посещаемыми местами в штате Небраска были Парк форта Робинсона, Исторический музей и парк Arbor Lodge, Carhenge (86 598), Музей пионеров прерии и исторический парк Ранчо Буффало Билла.	58%
Текст для просмотра. Текст из контрольного примера, оптимизированный с точки зрения удобства просмотра	В штате Небраска расположены знаменитые на весь мир ландшафты, которые неизменно притягивают к себе толпы людей. В 1996 году одними из самых популярных мест были: Парк форта Робинсона (355 000 посетителей), Исторический музей и парк Arbor Lodge (100 000), Carhenge (86 598), Музей пионеров прерии (60 000) Исторический парк Ранчо Буффало Билла (28 446)	47%

Окончание таблицы II.3.1

1	2	3
Объективный стиль. Использован нейтральный стиль без «рекламной воды» и преувеличений	В штате Небраска расположены несколько живописных ландшафтов, которые неизменно притягивают к себе толпы людей. В 1996 году одними из самых популярных мест были Парк форта Робинсона (355 000 посетителей), Исторический музей и парк Arbor Lodge (100 000), Carhenge (86 598), Музей пионеров прерии (60 000) и исторический парк Ранчо Буффало Билла (28 446)	27%
Комбинированная версия. Использованы все три улучшения стиля: – сжатость, – удобство просмотра, – объективность	В 1996 году в штате Небраска шестью самыми посещаемыми местами были: Парк форта Робинсона Исторический музей и парк Arbor Lodge Carhenge Музей пионеров прерии Исторический парк Ранчо Буффало Билла	124%

Авторы данного исследования разработали пять версий одного и того же сайта с различными формами представления информации, включая разные версии литредакции и форматирования текста при одной и той же навигации. В таблице II.3.1 показаны результаты измерений *юзабилити* (*usability*), полученные в результате тестирования работы с данным сайтом.

Интересно отметить, что юзабилити рекламного текста и нейтрального сильно отличаются. При чтении рекламного текста пользователям приходится тратить время не только на чтение, но и на отсеивание гипербол и выуживание реальных фактов.

4. Интернет-технологии в бизнесе

В недавнем прошлом основным режимом использования Интернета являлся режим электронной почты. Электронная почта – это абсолютно необходимое средство коммуникации, однако в ряде важных случаев (например, при необходимости оперативного поиска информации на серверах, подключенных к сети) оно не является достаточным: обмен информацией происходит слишком долго. В таких случаях требуется подключение к Интернету в режиме on-line. До недавнего времени такое подключение обходилось пользователям дороже, чем подключение в режиме электронной почты (расходуется больше телекоммуникационных ресурсов). Сейчас в мировом масштабе происходит скачкообразный переход на новые технологии телекоммуникаций, сопровождающийся резким увеличением пропускной способности каналов и таким же резким снижением стоимости их

использования. Это уже привело к тому, что в США режим on-line сегодня является основным режимом использования Интернета. Естественно, это привлекает бизнесменов (оперативный доступ к информации является очевидным требованием бизнеса).

При использовании Интернета в режиме on-line потенциально доступны многие программные сервисные средства, обеспечивающие:

– подключение к серверу в режиме удаленного терминала (Telnet);
– перекачку файлов (FTP);
– поиск необходимых информационных ресурсов и т.д.

Особенно важен тот факт, что потенциально любой пользователь, подключенный к Интернету в режиме on-line и обладающий IP-адресом, может создать свой собственный WWW-сервер, наполнив его актуальной информацией. Это открывает широкие возможности для бизнеса (реклама, каталоги и прайс-листы товаров и услуг, возможность дистанционных заказов и т.д.). Таким образом, не только информационная среда влияет на пользователя, но сам пользователь становится активным участником изменения среды (рис. II.4.1).

Многие коммерческие организации ранее мало использовали Internet-технологии из-за практически полного отсутствия безопасности информации при ее передаче по сети. По этой причине многие крупные компании, отделения которых располагаются в разных точках земного шара, до сих пор поддерживают собственные *корпоративные глобальные сети* с гарантированной безопасностью. Конечно, такие сети обходятся гораздо дороже, чем Интернет. В настоящее время начали появляться средства, обеспечивающие безопасность информации и при использовании Интернета.

Рис. II.4.1. Модель информационной гиперсреды

Например, компания *Sun Microsystems* объявила о выпуске продукта *SunScreen*, основанного на использовании методов криптографии на уровне передачи фрагментов сообщения. На основе применения SunScreen компания может создать виртуальную защищенную корпоративную подсеть внутри Интернета. Основной проблемой являются юридические ограничения применения методов криптографии, устанавливаемые национальными правительствами. Однако открывающиеся для бизнеса перспективы настолько заманчивы, что правительства международного сообщества будут вынуждены принять согласованное положительное решение.

На современном этапе развития электронных средств бизнеса можно выделить два основных направления использования Интернета в бизнесе:

– технологии Интернета в бизнесе,
– бизнес в Интернет-пространстве.

4.1. Технологии Интернета в бизнесе

В бизнесе Интернет используется практически с момента его зарождения. Любой компании необходимы информационное сопровождение своих бизнес-процессов, а также информационное взаимодействие в режиме on-line с внешней средой: филиалами в других городах и странах, клиентами, поставщиками – надежное и желательно недорогое. Те компании, которые первыми стали использовать электронную почту и телеконференции, на некоторое время получили конкурентное преимущество – развитые технологии позволяют практически мгновенно обмениваться качественной мультимедиа-информацией. Компании стали обзаводиться *информационными витринами* (сайтами), а многопрофильные компании и корпорации – *информационными порталами* (*Enterprise Information Portal – EIP*), которые очень быстро стали не только представлять «лицо» компании в бизнесе, но и превратились в один из мощных инструментов управления бизнесом.

Информационный портал представляет собой системную многоуровневую совокупность различных информационных ресурсов и сервисов организации, интегрирующую различные источники данных и отдельные функциональные системы, с единой точкой входа и унифицированными правилами представления и обработки информации.

С технологической точки зрения портал представляет собой сервер приложений, который может запускать стандартные «портальные» компоненты и гарантирует надежность и масштабируемость системы, а также берет на себя вопросы контроля прав доступа.

С точки зрения визуализации это отображающая часть информационной системы, обеспечивающая пользователей единым авторизованным персонифицированным доступом к внутренним и внешним информационным ресурсам и бизнес-приложениям.

С точки зрения реализации основной деятельности это новая концепция организации рабочих мест сотрудников с доступом ко всей информации, необходимой для выполнения ими предписанных функций.

С точки зрения управления организацией – интегрированная система управления распределенными информационными ресурсами и система информационного сопровождения всей деятельности организации.

Портал строится на базе Web-технологий, в его основе лежит *ядро*, обеспечивающее работу всех сервисов, интеграцию данных и приложений. Пользовательские функции реализуются посредством специализированных программных модулей – *портлетов*[1].

Создание и эффективное использование Web-порталов открывает принципиально новые возможности для использования Internet-технологий в бизнесе, позволяя:

– оперативно размещать и развивать информационные ресурсы организации;

– ускорить доступ к информации по тематике портала – в любой момент, в любой точке нахождения и для любого заинтересованного пользователя;

– повысить информативность лиц, занимающихся подготовкой принятия решения;

– формировать «клуб друзей организации» – заинтересовывать потенциальных заказчиков и клиентов качественными продуктами и услугами, системами скидок и бонусов, аккумулировать дополнительные финансовые ресурсы за счет привлекательных инвестиционных проектов и более активного использования информационных ресурсов организации широким кругом внешних пользователей;

– оптимизировать рекламный бюджет и ИТ-расходы организации (за счет организации Web-сервисов коллективного пользования);

[1] Портлет – подключаемый, сменный компонент пользовательского интерфейса веб-портала (элемент веб-страницы). Портлет выдает фрагменты разметки, которые встраиваются в страницу портала. Чаще всего страница портала представляется в виде набора не перекрывающих друг друга портлетных окон, каждое из которых отображает портлет. Таким образом, портлет (или совокупность портлетов) представляется в виде единого веб-приложения, размещенного на портале. Примеры портлетов: email, сообщения о погоде, последние новости. Благодаря существующим стандартам, разработчики могут создавать портлеты, встраиваемые в любой портал, следующий этим стандартам.

– интегрировать информационные ресурсы организации с ресурсами поставщиков, партнеров по бизнесу, мировыми информационными ресурсами;

– повысить качество управления процессами, информационной безопасностью и деятельностью организации в целом.

Преимущества, которые дают технологии Интернета в бизнесе:

1) *низкие затраты:*

– применение интернет-технологий для небольших и средних компаний существенно снижает затраты на создание, и главное – на эксплуатацию собственной распределенной корпоративной сети;

2) *открытость:*

– сетевые технологии являются полностью открытыми, потому что они основаны на стандартизированных и доступных каждому пользователю протоколах и форматах. Большое количество разработчиков прикладных пакетов осуществляет поддержку технологий в открытой среде. В связи с этим на рынке специализированного программного обеспечения достаточно много продуктов, что обеспечивает доступность и хороший выбор;

3) *устойчивость:*

– существует два критических фактора для успеха тех или иных технологий на рынке – надежность и масштабируемость. Internet/Intranet-технологии на сегодняшний день являются испытанными и надежными, так как они развиваются в течение длительного периода и используются миллионами людей во многих странах мира. Например, серверы компании Netscape фиксируют до 40 миллионов обращений в день;

4) *доступ к максимально широкой аудитории:*

– создав свою «виртуальную витрину» в World Wide Web, коммерческое предприятие получает доступ к любому заинтересованному пользователю и может напрямую взаимодействовать с потенциальными покупателями, предоставляя возможность полностью осуществить принцип «в любом месте в любое время»;

5) *снижение расходов на маркетинг и поддержку:*

– значительно уменьшаются расходы на традиционную рекламу, так как компания может размещать ее на собственном сайте в любых разумных количествах. Электронное распространение и поиск нужной информации обходится гораздо дешевле, чем на обычных бумажных носителях. При этом скорость распространения несравнимо выше. Электронную информацию можно постоянно обновлять, причем в автоматическом режиме. Всемерное распространение WWW открывает доступ практически в любой

уголок Земли, что в сочетании с технологиями электронной коммерции открывает путь на недостижимые до этого рынки;

6) *эффективное обеспечение работы компаний с распределенным производством:*

– многие компании имеют филиалы и подразделения в других районах и странах, где имеется избыток либо дешевой рабочей силы, либо других ресурсов. Информационные системы, включающие средства Интернета, позволяют осуществлять эффективное руководство разветвленными сетями производства и сбыта в режиме on-line;

7) *экономичное представление сотрудникам корпоративной и конфиденциальной информации:*

– внутрикорпоративные пространства Intranet с успехом используются как централизованные хранилища документов, с которыми постоянно работают сотрудники компании или с которыми руководство считает нужным их ознакомить. Intranet экономит время, устраняет необходимость изготовления и распространения печатных документов. Каждый сотрудник может обращаться к огромным массивам данных вне зависимости от того, где он находится и какую платформу он задействует.

4.2. Бизнес в Интернет-пространстве

Бизнес в Интернете основан на понимании того, что современный Интернет является сложившимся информационным виртуальным пространством, которое доступно любому пользователю Сети в любое время, в любой точке Земли. Любой клиент Интернета может автоматически стать частью этого виртуального мира, создав и предоставив другим пользователям новую частицу информации (см. рис. II.4.1).

Новейшие концепции и средства Интернета активно применяются при решении классических вопросов бизнеса: «Что делать?», «Где взять для этого средства?», «Кто есть кто?», «У кого купить и кому продать?», «Как это сделать, чтобы извлечь максимальную выгоду?». Очень существенны перспективы использования Интернета в банковском деле, в проведении маркетинга, при оказании услуг, при продажах, рекламе, аналитическом исследовании рынка, общении с поставщиками и заказчиками. Особая статья – отслеживание деятельности конкурентов и защита своей жизненно важной информации. Для этого особенно важны средства повышения безопасности информации в Сети.

Интерактивный характер взаимодействий в Интернете позволяет предоставлять виртуальные (но в то же время вполне реально доступные)

услуги: сетевые библиотеки, видеотеки, конференции, магазины и т.д. Возможность интерактивного взаимодействия позволяет пользователям, не выходя из офиса или дома, делать покупки в Интернет-магазинах, оплачивать услуги, играть на бирже, получать образование, повышать культурный уровень.

В настоящее время сформировались два понятия – *электронный бизнес* и *электронная коммерция*, которые при всем внешнем сходстве имеют существенные различия.

4.2.1. Электронный бизнес

Электронный бизнес (e-Business) означает осуществление и автоматизацию бизнес-процессов, а также повышение эффективности деятельности предприятия за счет повсеместного применения достижений из области Web-технологий. При этом фокус деловой активности перемещается на максимальное использование преимуществ внутренних и внешних связей компании в глобальных информационных сетях.

В электронном бизнесе можно выделить четыре слоя:

– Интернет-инфраструктура,
– Интернет-услуги,
– информационные посредники,
– электронная коммерция.

Инфраструктура реализуется телекоммуникационными компаниями и производителями программного обеспечения, компьютерного и сетевого оборудования. Услуги предоставляются Интернет-провайдерами, обеспечивающими *транзакции* в сети, и владельцами каналов связи. Инфраструктура услуг посредников включает службы, консультационные и обслуживающие компании, обеспечивающие создание Web-страниц и управление их содержанием (*Content Management System – CMS*), поисковые машины, базы данных и мультимедиа-применения. Каждый участник этого слоя активно способствует реализации электронной коммерции.

4.2.2. Электронная коммерция

Электронная коммерция (e-Commerce) является важнейшей составной частью электронного бизнеса. Это вид бизнеса, при котором взаимодействия (транзакции) между участниками коммерческих сделок происходят с помощью информационных технологий (электронные платежи, электронная цифровая подпись и пр.) или посредством Интернета.

Электронная коммерция, по сути, появилась раньше своего термина – в 1960-е годы, в эпоху *Mainframe-based приложений*. Одними из пер-

вых таких приложений были сервисные компьютерные программы для транспорта: заказ билетов, обмен данными между различными транспортными службами, подготовка и согласование маршрутов движения судов и самолетов.

Бурный рост сетей Интернета в 1990-е годы, связанный с появлением Web-технологий, заставил многих представителей бизнеса обратить пристальное внимание на его возможности. Появился новый тип бизнеса – розничная торговля через Интернет. В 1997 году появился стандарт *Open Buying on the Internet (OBI)*. В нем излагались принципы, которым должно соответствовать программное обеспечение для электронной коммерции, поддерживающее открытые Интернет-стандарты. Стандарт OBI затрагивает большой класс вопросов стандартизации всех форм взаимодействия между организациями, вовлеченными в полный цикл «Поставки – Продажи – Покупки» («Supply – Selling – Buying»).

В электронную коммерцию вовлекается все больше продавцов и покупателей. Обороты онлайновой торговли исчисляются к настоящему времени миллиардами долларов. Лидером электронной торговли является *Amazon.com*.

Электронная коммерция обладает рядом несомненных преимуществ, из которых можно выделить следующие:

– большая открытость компании по отношению к клиентам, взаимодействие с клиентами направлено на установление долгосрочных взаимоотношений (*Customer Relationship Management – CRM*);

– значительное увеличение оперативности получения информации для принятия решений – особенно при сложных торговых сделках с участием нескольких компаний;

– значительное сокращение цикла маркетинга и продаж, появление возможности пред- и послепродажной поддержки продукта – в особой степени это относится к программному обеспечению (представление подробной информации о продуктах и услугах, документация, поставка новых версий и т.д.);

– электронная оплата сделок с использованием электронных платежных систем;

– возможность организации виртуальных предприятий – группы отдельных специалистов или даже компаний для ведения совместной коммерческой деятельности;

– осуществление бизнес-процессов, совместно управляемых компанией и ее торговыми партнерами;

– значительное снижение затрат, связанных с обменом информацией, за счет использования более дешевых средств коммуникаций; возможность создавать альтернативные каналы продаж, например через электронный магазин на корпоративном портале;

– в случае необходимости факты и частота совершения торговых операций могут быть объективно измерены провайдером и подтверждены независимыми аудиторами, например, по анализу *log-файлов*;

– распространение права собственности на продаваемые или покупаемые нематериальные активы, например, пакеты информации в электронном виде;

– на рынке имеется достаточно недорогих программных пакетов для осуществления коммерческой деятельности в Интернете.

К недостаткам можно отнести:

– необходимость приобретения специализированных программно-аппаратных средств (если их нет в компании),

– осуществление повышенных мер безопасности информации,

– необходимость работы через Интернет-посредников (провайдеров),

– возможность потери критически важной для бизнеса информации.

Согласно стандарту OBI, электронной коммерцией считалось взаимодействие между бизнес-организациями посредством электронных технологий и Интернета. Сейчас это только один из секторов рынка электронной коммерции, который называется «Бизнес – бизнес» (*Business-to-Business*).

Электронную коммерцию в настоящее время принято разделять на несколько направлений (рис. II.4.2), основными из которых считаются:

– «Бизнес – бизнес» (*Business-to-Business – B2B*);

– «Бизнес – Потребитель» (*Business-to-Consumer, или Business-to-Customer, или Business-to-Client – B2C*);

– «Потребитель – Бизнес» (*Consumer-to-Business – C2B*);

– «Потребитель – Потребитель» (*Consumer-to-Consumer – C2C*).

Рассматриваются также взаимоотношения бизнеса и потребителей с государственными и иными регулирующими органами:

– «Бизнес – государственные органы» (*Business-to-Government – B2G*),

– «Потребитель – государственные органы» (*Consumer-to-Government – C2G*).

Можно выстраивать более сложные цепочки, такие как «Производитель электронного товара/услуги для государственного органа по социальному заказу – деятельность интернет-провайдера – государственный орган – потребитель» (*Business-to-Business-to-Government-to-Consumer – B2B2G2C*) и т.д.

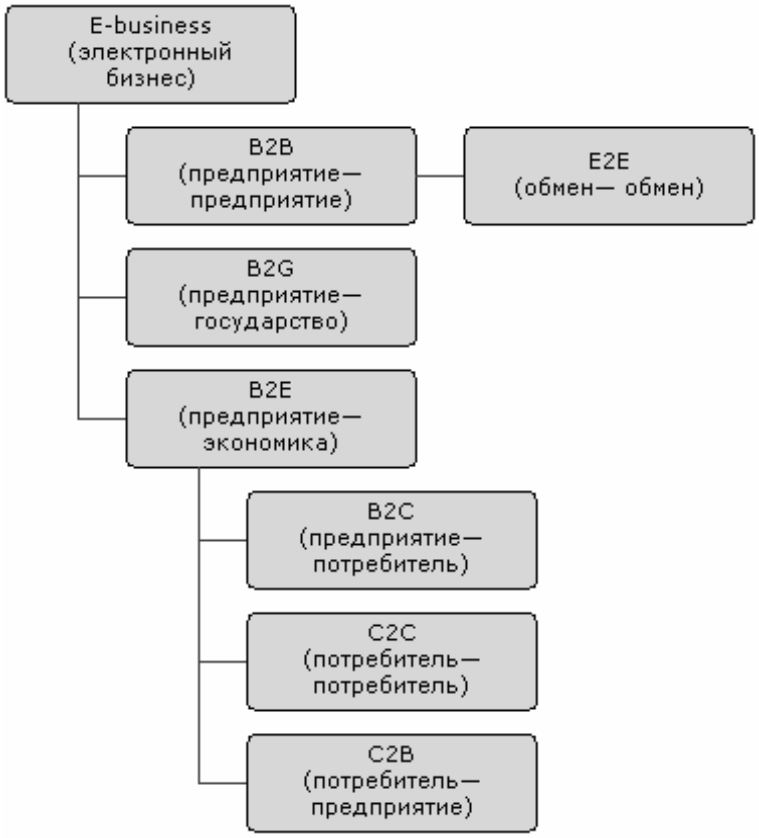


Рис. II.4.2. Направления электронной коммерции

Сектор B2B ранее определяли как межкорпоративное взаимодействие в системе «предприятие-предприятие» с использованием стандартов электронного обмена данными для осуществления передачи деловой информации. Изначально этим термином обозначались процессы купли-продажи товаров и услуг между предприятиями в режиме on-line. В настоящее время B2B понимается как любой процесс взаимодействия между предприятиями или подразделениями одного предприятия для решения бизнес-задач, который может быть реализован с применением информационных технологий и через Интернет. Полем деятельности участников этого сектора являются виртуальные *B2B-площадки* (рис. II.4.3).

Такие площадки принято делить на три типа в зависимости от того, кем они создаются:

– поставщиками, или продавцами (Supplier-driven, или Seller-driven);
– покупателями (Buyer-driven);
– третьей стороной (Third-party-driven).

Возникновение тех или иных видов торговых площадок зависит от степени влияния покупателей и продавцов в данной области экономики.

Рис. II.4.3. Виртуальные B2B-площадки

Площадки Supplier-driven. Крупные продавцы играют активную роль в формировании торговых площадок. Это происходит по разным причинам. Одни компании хотят привлечь как можно больше оптовых покупателей, другие – снизить затраты на продажи, третьи – иметь возможность объединиться с партнерами и диктовать свои условия на рынке.

Пример. 4.1. Несколько крупных американских компаний, выпускающих медицинские товары, – Johnson & Johnson, GE Medical Systems, Baxter International, Abbott Laboratories и Medtronic – предприняли усилия для создания общей Интернет-площадки Global Health Care Exchange (www.globalhc.com) в области здравоохранения для того, чтобы не платить комиссионные владельцам площадок, на которых они раньше работали. В России такие площадки организуются в сырьевой и обрабатывающей промышленности, в тяжелом машиностроении, в сельскохозяйственной отрасли, предприятиями энергетики и производителями продуктов питания.

Площадки Buyer-driven. Одна или несколько крупных компаний создают свою торговую площадку для привлечения компаний-поставщиков. Концепция таких торговых площадок возникла в связи с потребностями

крупных компаний в оптимизации процесса закупок, расширении торговых контактов и сети поставок по оптовым ценам. В качестве примера можно взять автомобильную промышленность США, где GM, Ford и Daimler Chrysler объединились для создания глобальной онлайновой торговой площадки, или здравоохранение, где Tenant Healthcare объединила усилия с Ventro для создания Internet-площадки, ориентированной на потребности рынка медицинских товаров.

Площадки Third-party-driven. Такие площадки создаются, обслуживаются и управляются третьей стороной для того, чтобы свести вместе покупателей и продавцов. Обычно такие площадки создаются теми, кто хорошо ориентируется в том или ином секторе бизнеса, и служат для получения дохода в виде процента от совершенных сделок. Примерами таких площадок могут быть *электронные биржи и аукционы* (*B2B Exchange/Auction*).

При формировании площадок B2B необходимо учитывать ряд важных аспектов:

– доступность для новых участников;

– поддержка признанных стандартов разработки (EDI, Web-формы, XML-приложения);

– масштабируемость используемых платформ;

– возможность управления информацией и применения аналитических методов обработки;

– возможность интеграции инструментов электронной коммерции;

– обеспечение информационной безопасности.

В зависимости от конкретного типа площадки делают акцент на те или иные характеристики и разрабатывают соответствующие инструменты для поставщиков либо для потребителей.

Сектор B2C – это форма электронной коммерции, целью которой являются прямые продажи для потребителя. Такая форма торговли эффективна для устранения географической удаленности между крупными городами и регионами в смысле доступности товаров и услуг для потребителя. B2C позволяет вести прямые продажи с минимальным количеством посредников. Устранение посредников дает возможность устанавливать конкурентные цены на местах и даже увеличивать их (исключая процент посредников), что естественно приводит к росту прибыли.

К системам B2C относятся:

– Web-витрины (Front Office) торговых компаний для привлечения возможных покупателей к продуктам данных компаний;

– интернет-магазины, которые занимаются только продажей товаров и содержат необходимую инфраструктуру (*Back Office*) для производства, продаж и управления электронной торговлей через Интернет;

– торговые интернет-компании, в которых система электронных продаж (Back Office) полностью интегрирована со всеми торговыми бизнес-процессами.

Для полноценного функционирования Internet-магазина необходимы следующие обязательные компоненты:

– Web-сервер, производящий разграничение доступа и распределяющий запросы;

– сервер приложений, управляющий бизнес-логикой и реализующий необходимую совокупность процессов;

– БД и СУБД для сбора, хранения, обработки и управления данными;

– система электронных платежей, включающая электронную цифровую подпись.

Структура управления интернет-магазином реализуется, как правило, в виде трехзвенной архитектуры «клиент – сервер приложений – сервер базы данных». Для интеграции интернет-магазина с бизнес-процессами основной компании может быть установлен *шлюз-конвертор*, который будет передавать данные от магазина в бухгалтерскую систему и систему документационного обеспечения компании.

Сектор C2B. Во-первых, C2B – это форма электронной коммерции, которая предоставляет потребителю возможность самостоятельно устанавливать цену на различные товары и услуги, предлагаемые компаниями. Таким способом формируется спрос, который, однако, не означает, что продажа совершится по запрошенной цене. Продавец, пользуясь статистическими данными текущего спроса, принимает окончательное решение, и после этого товар «выпускается» в продажу по усредненной цене.

Во-вторых, C2B – совокупность методов, инструментов и технологий для выполнения онлайновых транзакций между потребителями (физическими лицами или небольшими объединениями частных предпринимателей) и предприятиями. Примером являются сайты бизнес-консультантов, юристов, промоутеров, профитеров (специалистов по оптимизации деятельности предприятий), аудиторов, рекламных агентов и других специалистов, способных оказывать услуги предприятиям.

Сектор C2C – это форма электронной торговли, суть которой состоит в организации купли-продажи товаров и услуг между потребителями. В этом случае персональный сайт физического лица или специализированный сайт, имеющий раздел бесплатных объявлений, выступает в роли по-

средника между покупателем и продавцом. Сделка может быть совершена как непосредственно через Интернет, если обе стороны имеют платежные инструменты, так и наличными деньгами при согласовании всех вопросов с применением Интернета. В качестве примера можно привести ресурс www.molotok.ru – один из ведущих российских аукционов, где каждый желающий может что-либо продать или купить.

В заключение отметим, что сумма всех четырех секторов электронной коммерции «B2B + B2C + C2B + C2C» является ее обобщенным социальным ресурсом. Электронный бизнес в настоящее время насчитывает много разновидностей, определяемых конкретными задачами бизнеса и применяемыми информационными технологиями. Это торговые интернет-системы, интернет-биржи и аукционы, электронные платежи, кредитные и дебетовые системы, электронные чеки и деньги, интернет-страхование, аренда Web-сервисов, подбор персонала, лингвистические услуги и многое другое. В зависимости от сфокусированности контента ресурс «B2B + B2C + + C2B + C2C» реализует соответствующий аспект электронного бизнеса переходного периода – от постиндустриального общества к информационному.

4.2.3. Мобильная коммерция

Мобильная коммерция (m-Commerce) – это перевод электронной коммерции в мобильные формы.

Мобильная коммерция делает пользователя еще более независимым, не привязывая его к стационарному компьютеру или серверу. Она предоставляет все возможности e-Commerce, при наличии у пользователя мобильных устройств для получения и передачи информации, выхода в Интернет и совершения транзакций.

Основу мобильной коммерции составляют протоколы WAP и GPRS, позволяющие выходить в Сеть, скачивать информацию, просматривать ее на мини-дисплее и совершать многочисленные операции электронной коммерции. Мобильная коммерция развивается очень быстрыми темпами.

Пример. 4.2. Всем известный мобильный телефон с функцией WAP или собственным микробраузером становится сейчас еще и средством идентификации владельца, выполняет функции банковской кредитной карты, позволяет выходить в мировую сеть, заказывать билеты, производить коммерческие операции и т.д.

Развитие Всемирной сети и взаимодействие региональных сетей приведет к полному преобразованию бизнеса и новый WWW-бизнес станет более выгодным и более удобным для людей.

5. Облачные вычисления

Облачные вычисления (англ. *cloud computing*) в информатике – это модель обеспечения повсеместного и удобного сетевого доступа по требованию к общему пулу конфигурируемых вычислительных ресурсов (например, сетям передачи данных, серверам, устройствам хранения данных, приложениям и сервисам – как вместе, так и по отдельности), которые могут быть оперативно предоставлены и освобождены с минимальными эксплуатационными затратами и/или обращениями к провайдеру.

Приведем основные характеристики облачных вычислений:

– *самообслуживание по требованию* (*англ. self service on demand*), потребитель самостоятельно определяет и изменяет вычислительные потребности, такие как серверное время, скорости доступа и обработки данных, объем хранимых данных без взаимодействия с представителем поставщика услуг;

– *универсальный доступ по сети*, услуги доступны потребителям по сети передачи данных вне зависимости от используемого терминального устройства;

– *объединение ресурсов* (*англ. resource pooling*), поставщик услуг объединяет ресурсы для обслуживания большого числа потребителей в единый пул для динамического перераспределения мощностей между потребителями в условиях постоянного изменения спроса на мощности; при этом потребители контролируют только основные параметры услуги (например, объем данных, скорость доступа), но фактическое распределение ресурсов, предоставляемых потребителю, осуществляет поставщик (в некоторых случаях потребители все-таки могут управлять некоторыми физическими параметрами перераспределения, например, указывать желаемый центр обработки данных из соображений географической близости);

– *эластичность*, услуги могут быть предоставлены, расширены, сужены в любой момент времени, без дополнительных издержек на взаимодействие с поставщиком, как правило, в автоматическом режиме;

– *учет потребления*, поставщик услуг автоматически исчисляет потребленные ресурсы на определенном уровне абстракции (например, объем хранимых данных, пропускная способность, количество пользователей, количество транзакций), и на основе этих данных оценивает объем предоставленных потребителям услуг.

5.1. Модели развертывания

5.1.1. Частное облако

Частное облако (*англ. private cloud*) – инфраструктура, предназначенная для использования одной организацией, включающей несколько потребителей (например, подразделений одной организации), возможно также клиентами и подрядчиками данной организации. Частное облако может находиться в собственности, управлении и эксплуатации как самой организации, так и третьей стороны (или какой-либо их комбинации), и она может физически существовать как внутри так, и вне юрисдикции владельца.

5.1.2. Публичное облако

Публичное облако (англ. *public cloud*) – инфраструктура, предназначенная для свободного использования широкой публикой. Публичное облако может находиться в собственности, управлении и эксплуатации коммерческих, научных и правительственных организаций (или какой-либо их комбинации). Публичное облако физически существует в юрисдикции владельца – поставщика услуг

5.1.3. Гибридное облако

Гибридное облако (англ. *hybrid cloud*) – это комбинация из двух или более различных облачных инфраструктур (частных, публичных или коммунальных), остающихся уникальными объектами, но связанных между собой стандартизованными или частными технологиями переносимости данных и приложений (например, кратковременное использование ресурсов публичных облаков для балансировки нагрузки[1] между облаками).

5.1.4. Общественное облако

Общественное облако (англ. *community cloud*) – вид инфраструктуры, предназначенный для использования конкретным сообществом потребителей из организаций, имеющих общие задачи (например, миссии требований безопасности, политики, и соответствия различным требованиям). Общественное облако может находиться в кооперативной (совместной) собственности, управлении и эксплуатации одной или более из организаций сообщества или третьей стороны (какой-либо их комбинации), и она может физически существовать как внутри, так и вне юрисдикции владельца.

[1] В терминологии компьютерных сетей балансировка (выравнивание) нагрузки – распределение процесса выполнения заданий между несколькими серверами сети с целью оптимизации использования ресурсов и сокращения времени вычисления.

5.1.5. Экономические аспекты использования облачных вычислений

Использование облачных вычислений позволяет:

– существенно снизить капитальные расходы на построение центров обработки данных, закупку серверного и сетевого оборудования, аппаратных и программных решений по обеспечению непрерывности и работоспособности;

– практически мгновенно реагировать на увеличение спроса на вычислительные мощности.

5.1.6. Облачное хранилище данных. Облачные шлюзы

Облачное хранилище данных – модель онлайн-хранилища, в котором данные хранятся на многочисленных, распределенных в сети серверах, предоставляемых в пользование клиентам, в основном третьей стороной. В противовес модели хранения данных на собственных, выделенных серверах, приобретаемых или арендуемых специально для подобных целей, количество или какая-либо внутренняя структура серверов клиенту, в общем случае, не видна.

Данные хранятся и обрабатываются в так называемом облаке, которое представляет собой, с точки зрения клиента, один большой, виртуальный сервер. Физически же такие сервера могут располагаться весьма удаленно друг от друга географически, вплоть до расположения на разных континентах.

Преимущества использования облачных хранилищ:

– клиент платит только за то место в хранилище, которое фактически использует, но не за аренду сервера, все ресурсы которого он может и не использовать.

– клиенту нет необходимости заниматься приобретением, поддержкой и обслуживанием собственной инфраструктуры по хранению данных, что, в конечном счете, уменьшает общие издержки производства.

– все процедуры по резервированию и сохранению целостности данных производятся провайдером облачного центра, который не вовлекает в этот процесс клиента.

Однако с использованием облачных вычислений потенциально связаны следующие вопросы:

– безопасность при хранении и пересылке данных, особенно в отношении конфиденциальных и приватных данных;

– общая производительность при работе с данными (может быть ниже, чем при работе с локальными копиями данных);

– надежность и своевременность получения и доступности данных в облаке (зависит от многих промежуточных параметров, в основном таких как каналы передачи данных на пути от клиента к облаку

– вопрос последней мили;

– вопрос о надлежащем качестве работы интернет-провайдера клиента;

– вопрос о доступности самого облака в данный момент времени.

Облачные шлюзы – технология, которая может быть использована для более удобного представления облака клиенту. К примеру, с помощью соответствующего программного обеспечения, хранилище в облаке может быть представлено для клиента как локальный диск на компьютере. Таким образом, работа с данными в облаке для клиента становится абсолютно прозрачной. И при наличии хорошей, быстрой связи с облаком клиент может даже не замечать, что работает не с локальными данными у себя на компьютере, а с данными, хранящимися, возможно, за много сотен километров от него.

Приведем наиболее крупных провайдеров облачных хранилищ:

– 4shared;
– Amazon S3;
– Box.net;
– Dropbox;
– iWork.com;
– Google Docs;
– SpiderOak;
– Syncplicity;
– SugarSync;
– Ubuntu One;
– Windows Live SkyDrive;
– Wuala;
– ZumoDrive.

5.2. Модели обслуживания

5.2.1. Программное обеспечение как услуга

Программное обеспечение как услуга (*SaaS*, англ. *Software-as-a-Service*) – модель, в которой потребителю предоставляется возможность использования прикладного программного обеспечения провайдера, работающего в облачной инфраструктуре и доступного из различных клиентских устройств или посредством тонкого клиента, например, из браузера (например, веб-почта) или интерфейс программы. Контроль и управление

основной физической и виртуальной инфраструктурой облака, в том числе сети, серверов, операционных систем, хранения, или даже индивидуальных возможностей приложения (за исключением ограниченного набора пользовательских настроек конфигурации приложения) осуществляется облачным провайдером.

5.2.2. Платформа как услуга

Платформа как услуга (*PaaS*, англ. *Platform-as-a-Service*) – модель, когда потребителю предоставляется возможность использования облачной инфраструктуры для размещения базового программного обеспечения для последующего размещения на нем новых или существующих приложений (собственных, разработанных на заказ или приобретенных тиражируемых приложений). В состав таких платформ входят инструментальные средства создания, тестирования и выполнения прикладного программного обеспечения, – системы управления базами данных, связующее программное обеспечение, среды исполнения языков программирования – предоставляемые облачным провайдером.

Контроль и управление основной физической и виртуальной инфраструктурой облака, в том числе сети, серверов, операционных систем, хранения осуществляется облачным провайдером, за исключением разработанных или установленных приложений, а также, по возможности, параметров конфигурации среды (платформы).

5.2.3. Инфраструктура как услуга

Инфраструктура как услуга (*IaaS*, англ. *IaaS or Infrastructure-as-a-Service*) предоставляется как возможность использования облачной инфраструктуры для самостоятельного управления ресурсами обработки, хранения, сетями и другими фундаментальными вычислительными ресурсами, например потребитель может устанавливать и запускать произвольное программное обеспечение, которое может включать в себя операционные системы, платформенное и прикладное программное обеспечение. Потребитель может контролировать операционные системы, виртуальные системы хранения данных и установленные приложения, а также может иметь ограниченный контроль набора доступных сервисов (например, межсетевой экран, DNS). Контроль и управление основной физической и виртуальной инфраструктурой облака, в том числе сети, серверов, типов используемых операционных систем, систем хранения осуществляется облачным провайдером.

Словарь терминов

DNS (Domain Name System) – служба имен доменов для перевода имен компьютеров в сети Интернет в их IP-адреса.

E-Mail – сетевая служба, позволяющая обмениваться текстовыми электронными сообщениями через Интернет.

FTP (File Transfer Protocol) – протокол передачи файлов.

HTML (HyperText Markup Language) – язык гипертекстовой разметки для создания Web-документов.

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста.

ICQ – служба поиска сетевого *IP*-адреса человека, подключенного в данный момент к Интернету.

IP (Internet Protocol) – протокол межсетевого взаимодействия, обеспечивает маршрутизацию пакетов в сети.

LAN (Local Area Network) – локальная компьютерная сеть.

MAN (Metropolitan Area Network) – региональная компьютерная сеть.

TCP (Transmission Control Protocol) – протокол контроля передачи информации в сети. TCP – протокол транспортного уровня, один из основных протоколов сети Интернет.

TCP/IP (Transmission Control Protocol/Internet Protocol) – технология межсетевого взаимодействия.

Telnet – сетевая служба, предоставляющая возможность абоненту работать на любой ЭВМ сети Интернет, как на своей собственной (удаленный доступ).

URL-адрес – сетевое расширение понятия полного имени ресурса в операционной системе.

Usenet – служба телеконференций.

WAN (Wide Area Network) – глобальная компьютерная сеть.

Web-канал – Web-узел, способный самостоятельно инициировать поставку информации.

Web-навигация – перемещение между Web-документами.

Web-страница – отдельный документ WWW, как правило написанный на языке HTML. Web-страница может содержать текст, графику, звуковое сопровождение, анимацию и другие мультимедийные объекты, а также гиперссылки.

Web-узел – группа тематически объединенных Web-страниц, которые обычно связаны гиперссылками.

World Wide Web – это единое информационное пространство, состоящее из взаимосвязанных электронных документов, хранящихся на Web-серверах.

Брандмауэр – используется для обеспечения сетевой безопасности между локальной и глобальной сетью. Брандмауэром может быть специальный компьютер или компьютерная программа, препятствующая несанкционированному перемещению данных между сетями.

Браузер – программа для просмотра Web-страниц (например, Internet Explorer, Netscape Navigator и др.).

Гибридное облако – это комбинация из двух или более различных облачных инфраструктур, остающихся уникальными объектами, но связанных между собой стандартизованными или частными технологиями переносимости данных и приложений.

Гиперссылка – это выделенный фрагмент документа (текст или иллюстрация), с которым ассоциирован адрес другого Web-документа.

Домен – определяет имя некоторой части сети Интернет.

Интернет – это глобальная компьютерная сеть, состоящая из множества соединенных друг с другом меньших по размеру сетей и покрывающая весь земной шар.

Интернет-провайдер – это организация, которая имеет собственную высокоскоростную сеть, объединенную с другими сетями по всему земному шару.

Интерфейс – правила и стандартизованные форматы сообщений, обеспечивающие взаимодействие модулей, реализующих протоколы соседнего уровня и находящихся в одном компьютере.

Информационный портал – это системная многоуровневая совокупность различных информационных ресурсов и сервисов организации, интегрирующая различные источники данных и отдельные функциональные системы, с единой точкой входа и унифицированными правилами представления и обработки информации.

Клиент сети – любой компьютер, имеющий доступ к услугам сервера.

Коммутация – это способ соединения абонентов, который обеспечивает разделение имеющихся физических каналов между несколькими сеансами связи и между абонентами сети.

Коммутация каналов – подразумевает образование составного канала из последовательно соединенных отдельных канальных участков для прямой передачи данных между узлами сети.

Коммутация пакетов – была специально разработана для эффективной передачи компьютерного трафика в сетях, где различные компьютеры могут иметь различное быстродействие.

Коммутация сообщений – это передача единого блока данных через промежуточные транзитные компьютеры с временной буферизацией этого блока на диске каждого компьютера.

Маршрутизатор – компьютер сети, занимающийся выбором кратчайшего маршрута следования пакетов в сети.

Мобильная коммерция – это перевод электронной коммерции в мобильные формы.

Облачное хранилище данных – это модель онлайн-хранилища, в котором данные хранятся на многочисленных, распределенных в сети серверах, предоставляемых в пользование клиентам, в основном третьей стороной.

Облачные вычисления – это модель обеспечения повсеместного и удобного сетевого доступа по требованию к общему пулу конфигурируемых вычислительных ресурсов, которые могут быть оперативно предоставлены и освобождены с минимальными эксплуатационными затратами и/или обращениями к провайдеру.

Общественное облако – вид инфраструктуры, предназначенный для использования конкретным сообществом потребителей из организаций, имеющих общие задачи.

Протоколы – это формализованные правила, определяющие последовательность и формат сообщений, которыми обмениваются модули, лежащие на одном уровне, но в различных компьютерах.

Публичное облако – это инфраструктура, предназначенная для свободного использования широкой публикой.

Релевантный документ – это документ, содержащий искомую информацию.

Сектор B2B – это межкорпоративное взаимодействие в системе «предприятие-предприятие» с использованием стандартов электронного обмена данными для осуществления передачи деловой информации.

Сектор B2C – это форма электронной коммерции, целью которой являются прямые продажи для потребителя.

Сектор C2B – это форма электронной коммерции, которая предоставляет потребителю возможность самостоятельно устанавливать цену на различные товары и услуги, предлагаемые компаниями.

Сектор C2C – это форма электронной торговли, суть которой состоит в организации купли-продажи товаров и услуг между потребителями.

Сервер сети – компьютер, управляющий обменом данных по сети и распределением ресурсов.

Служба Интернета – пара программ, взаимодействующих между собой согласно определенным правилам, называемым протоколами.

Стек коммуникационных протоколов – это иерархически организованный набор протоколов для взаимодействия компьютеров в сети.

Топология сети – это геометрическая форма и физическое расположение компьютеров по отношению друг к другу.

Точность поиска – это доля релевантных документов в списке всех найденных поисковой машиной документов.

Тэг – элемент языка разметки гипертекста. Для того чтобы тэги отличались от обычного текста, их заключают в угловые скобки (< и >).

Хост – это объект сети, который может передавать и принимать IP-пакеты.

Частное облако – это инфраструктура, предназначенная для использования одной организацией, включающей несколько потребителей, возможно также клиентами и подрядчиками данной организации.

Электронная коммерция – это вид бизнеса, при котором транзакции между участниками коммерческих сделок происходят с помощью информационных технологий или посредством Интернета.

Электронный бизнес – это осуществление и автоматизация бизнес-процессов, а также повышение эффективности деятельности предприятия за счет повсеместного применения достижений из области Web-технологий.

Вопросы и задания для самоконтроля

1. Каково назначение компьютерных сетей?
2. Перечислите и охарактеризуйте три типа сетевых ресурсов.
3. По каким признакам классифицируются компьютерные сети?
4. Какова классификация компьютерных сетей по территориальной распространенности?
5. На какие группы делятся компьютерные сети по скорости передачи информации?
6. Что такое топология сети?
7. Классифицируйте компьютерные сети по топологии.
8. Для чего предназначен сетевой адаптер?
9. Какие виды кабеля используют для конструирования сетей?
10. Классифицируйте компьютерные сети по способу управления.
11. Классифицируйте компьютерные сети по методу коммутации.
12. Что представляет собой маршрутизатор?
13. Что такое брандмауэр?
14. В чем состоит сущность стандартизации компьютерных сетей?
15. Охарактеризуйте сетевые протоколы TCP, IP, TCP/IP.
16. Сколько классов сетей определяет IP-адресация, перечислите их и дайте характеристику каждому.
17. Что такое интерфейс?
18. Что представляет собой глобальная сеть Интернет?
19. Как происходило развитие глобальной сети интернет?
20. Что такое домен?
21. Что такое интернет-провайдер?

22. Что представляют собой точки присутствия интернет-провайдера?

23. Как объединяют своих клиентов в одну сеть различные интернет-провайдеры?

24. Охарактеризуйте основные службы Интернета.

25. Каким образом осуществляется адресация компьютеров в Интернете?

26. Кто является основоположником Всемирной паутины?

27. Каково назначение браузера?

28. Что такое Web-страница, Web-узел?

29. Для чего используются гиперссылки?

30. Как называется процесс перемещения между Web-документами с помощью гипертекстовых ссылок?

31. Охарактеризуйте различные способы поиска информации в Интернете.

32. Что такое релевантный документ?

33. Что означает аббревиатура HTML?

34. Что такое тэг? Какие виды тэгов существуют?

35. Для чего используется пара тэгов <html> … </html>?

36. Для чего используется пара тэгов <head> … </head>?

37. Для чего используется пара тэгов <title> … </title>?

38. Для чего используется пара тэгов <body> … </body>?

39. Какая пара тэгов используется для создания гиперссылок? Назовите и охарактеризуйте основные атрибуты этой пары тэгов.

40. Назовите и охарактеризуйте два основных направления использования Интернета в бизнесе.

41. Что представляет собой информационный портал?

42. Что такое портлет?

43. Перечислите новые возможности для использования Интернет-технологий в бизнесе.

44. Какие преимущества предоставляет использование технологий Интернета в бизнесе?

45. Назовите и охарактеризуйте четыре слоя электронного бизнеса.

46. Что такое электронная коммерция? Каковы достоинства и недостатки использования мобильной коммерции?

47. Назовите и охарактеризуйте основные направления электронной коммерции.

48. Что представляют собой облачные вычисления?

49. Перечислите основные характеристики облачных вычислений.

50. Охарактеризуйте общественное, частное, публичное и гибридное облако.

51. Что представляет собой облачное хранилище данных?

52. Охарактеризуйте различные модели обслуживания.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

ЛАБОРАТОРНАЯ РАБОТА № 1
Пример создания Windows-приложения в VBA для Microsoft Excel

Цель работы: знакомство с элементами ActiveX (элементами управления) и компонентами редактора Visual Basic for Applications, а также с этапами создания Windows-приложения.

Теоретические сведения. См. разделы 2.3 и 3.2 лекций.

Процесс создания Windows-приложения состоит из следующих этапов:

1. *Постановка задачи* – составление по возможности точного и понятного словесного описания того, как должно работать будущее приложение, – что должен делать пользователь в процессе его работы. Это описание должно объяснять и то, как будет представлена информация, которую нужно преобразовать с помощью этого приложения, например исходные данные и результаты.
2. *Разработка интерфейса* – создание экранной формы (окна приложения) со всеми находящимися на этой форме объектами и свойствами этих объектов.
3. *Собственно программирование* – определение того, какие события будут происходить в процессе работы приложения, составление алгоритмов процедур для этих событий и написание программы (программных кодов этих процедур).
4. *Отладка программы* – устранение логических ошибок в процедурах и достижение удовлетворительной работы приложения в среде проектирования.
5. *Сохранение проекта* и, при желании, *компиляция* – превращение проекта в исполняемое приложение, способное работать самостоятельно (за пределами среды проектирования).

Постановка задачи

Все этапы проектирования Windows-приложения в среде Visual Basic for Applications будем рассматривать на примере.

Пример. Необходимо произвести ремонт комнаты. Определить с помощью компьютера нужное количество необходимых для ремонта материалов (например, обоев для оклейки стен). Для того чтобы это сделать нужно вычислить площадь всех стен комнаты, имеющей форму прямо-

угольного параллелепипеда. (Зная площадь стен, можно купить нужное количество рулонов обоев.)

Прежде всего выбираем приложению имя, например *Площадь стен комнаты*.

Описываем то, как пользователь будет работать с данным приложением. Это описание может выглядеть следующим образом:

– для нахождения площади стен понадобятся размеры комнаты. Эти размеры – длину, ширину и высоту – пользователь приложения будет задавать в процессе его работы.

Пользователь после запуска приложения увидит на экране окно, типичное для приложений ОС Windows. Такое окно должно содержать стандартные элементы этой системы – кнопки минимизации, развертывания и закрытия окна, строку заголовка.

Кроме стандартных элементов в окне приложения должны быть и нестандартные элементы, созданные программистом. Прежде всего строка заголовка должна содержать название приложения: *Площадь стен комнаты*.

Слева от названия может находиться пиктограмма – особый значок для обозначения данного приложения.

Для наглядности в окне приложения может находиться чертеж или рисунок комнаты. Чертеж или рисунок – это графический элемент приложения, который тоже может быть создан программистом. Другие нестандартные элементы необходимы для того, чтобы приложение работало в соответствии с замыслами его создателя.

Исходные данные пользователь будет вводить с помощью клавиатуры в специальные небольшие окошки (текстовые поля), находящиеся в окне приложения (на экранной форме). Очевидно, что в нашем примере таких нестандартных элементов должно быть 3 – по числу вводимых размеров комнаты.

После ввода исходных данных пользователь с помощью мыши щелкнет командную кнопку, которая также находится на экранной форме. Она также представляет собой нестандартный элемент. Незамедлительно в специальном маленьком окошке (еще одном текстовом поле) должен появиться результат. После этого пользователь может закончить работу приложения или ввести какие-нибудь другие исходные данные. И опять щелкнуть командную кнопку.

Исходные данные пользователь будет вводить с такой точностью, которая его устраивает, например, `4.05, 3.25, 2.55` – соответственно, длина, ширина и высота комнаты в метрах.

Ответ (результат) будет выдаваться также с необходимой точностью, например, `37.23` – площадь стен комнаты в квадратных метрах.

Чтобы пользователь знал, для каких именно данных предназначено то или иное текстовое поле, рядом с каждым из них будет стоять метка – пояснительная надпись. Метки – нестандартные элементы, создаваемые программистом. С помощью особой метки пользователю будет показана формула, с помощью которой компьютер вычисляет площадь стен, например: `S = = 2 (A + B) H`, где `A`, `B` и `H` – длина, ширина и высота комнаты.

Будем считать, что на этом этап постановки задачи создания Windows-приложения «Площадь стен комнаты» окончен.

Решение

Создание экранной формы (разработка интерфейса)

Рассмотрим следующий этап процесса проектирования приложения – создание экранной формы. Этот этап также называют *разработкой интерфейса*.

Создание экранной формы – это не менее важная стадия проектирования, чем программирование, которому эта стадия предшествует.

Разработка интерфейса состоит из следующих шагов:

– создание эскиза экранной формы;

– вход в среду проектирования Visual Basic for Applications;

– создание экранной формы и установка значений свойств этой формы;

– создание на форме объектов управления и установка значений свойств этих объектов.

Создание эскиза экранной формы. Как следует из постановки задачи, необходимо создать окно Windows-приложения, в котором будет 4 текстовых поля: 3 из них для ввода исходных данных – длины, ширины и высоты комнаты, и одно для вывода результата – площади стен этой комнаты.

Кроме текстовых полей, на экранной форме должны быть все необходимые для пользователя пояснения (в виде заголовка, надписей, расчетной формулы и даже небольшого чертежа).

Необходимо также разместить командную кнопку. Ее нажатие должно служить сигналом для расчета по формуле и выдачи на экран результата (рис. Л1.1).

Рис. Л1.1. Эскиз экранной формы (окна приложения)

Вход в среду проектирования Visual Basic for Applications

К работе на компьютере можно приступать, когда эскиз окна будущего приложения готов,

Вход в среду проектирования Visual Basic for Applications можно выполнить либо по команде *меню Сервис → Макрос → Редактор Visual Basic for Application*, либо нажатием сочетания клавиш Alt+F11.

Создание экранной формы и установка значений свойств этой формы

После того как откроется окно редактора Visual Basic, здесь следует открыть окно экранной формы по команде *меню Insert → UserForm* (рис. Л1.2). По умолчанию она имеет заголовок `UserForm1`.

В языке Visual Basic for Applications рассматривается такое понятие как класс объектов. В частности, рассматривается класс объектов `Form` (`Экранная форма`). Это совокупность общих характеристик любой экранной формы. Когда создается конкретная экранная форма или какой-нибудь иной конкретный объект, можно использовать уже имеющийся в системе список свойств класса этих объектов. Именно этот список находится в левой части рисунка Л1.2. – в окне свойств (`Properties`).

Если окно свойств не отображается, его можно открыть по команде *меню View → Properties Window* или по нажатию функциональной клавиши F4.

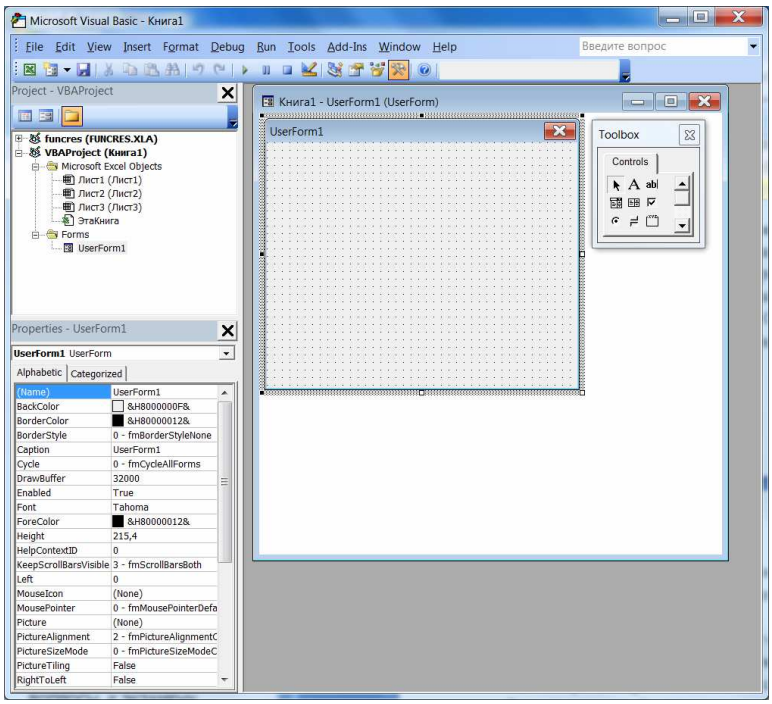


Рис. Л1.2. Окно редактора Visual Basic for Application на начальной стадии проектирования интерфейса

Прежде всего установим значения размеров формы – свойств Width (Ширина) и Height (Высота). Эти значения можно установить в окне Properties, внося конкретные числа в правую колонку таблицы, находящейся в этом окне. Но лучше увеличить мышью размеры экранной формы. При этом значения в окне свойств будут изменяться автоматически.

Значения свойств положения формы на экране – Left (Левый край) и Top (Верхний край) – тоже легко установить с помощью мыши.

А вот для установки значений имени (идентификатора) формы, надписи в строке заголовка формы, цвета фона формы будет необходимо окно свойств.

Этими свойствами соответственно являются: (Name), Caption, BackColor. Значения первых двух из этих свойств установим такими: FПлощадьСтен, Площадь стен комнаты. Эти значения набираются с клавиатуры. Значение свойства BackColor выбирается с помощью раскрывающегося списка с двумя вкладками (рис. Л1.3) после щелчка на кнопке ▾ в строке *BackColor* окна свойств.

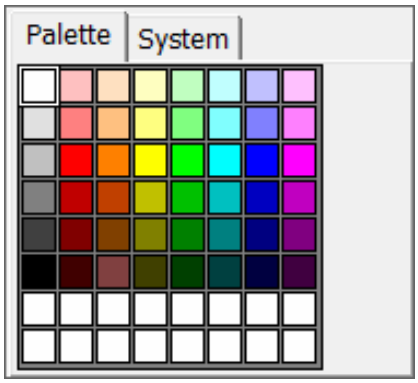


Рис. Л1.3. Раскрывающийся список для установки значения свойства BackColor экранной формы

В этом списке щелчком мыши выбираем вкладку «Palette», а в появившейся палитре выбираем образец белого цвета – соответственно форма окрасится в белый цвет.

Создание на экранной форме объектов управления и установка значений свойств этих объектов

На экранной форме необходимо разместить следующие объекты: 6 объектов класса Метка, 4 объекта класса Текстовое поле, по одному объекту классов Изображение и Командная кнопка. Для их создания будет использоваться «Окно инструментов» Toolbox. Если это окно не отображается, то следует его открыть, выполнив команду *меню View → → Toolbox*.

Размещение объектов Метка. Все надписи, заголовок и формула будут размещены в нескольких таких объектах.

В «Окне инструментов» находятся инструменты, с помощью которых можно создать любое количество объектов управления одного и того же класса. В частности, с помощью инструмента Label можно создать любое количество объектов Метка.

 – пиктограмма этого инструмента.

Рассмотрим технологию создания меток на примере размещения на экранной форме одной из них:

1) щелчком мыши выбрать пиктограмму Label в окне Toolbox;

2) поместить указатель мыши в то место экранной формы, где будет находиться левый верхний угол будущего объекта. При нажатой левой кнопке мыши «протащить» указатель в то место, где будет находиться правый нижний угол будущего объекта. Этот процесс будет сопровождаться растяжением пунктирной рамки – границы создаваемого объекта.

Размеры и положение появившегося прямоугольника можно подкорректировать с помощью мыши.

Остальные 5 объектов Метка необходимо создать таким же способом.

Размещение текстовых полей. Размещение текстовых полей выполняется с помощью инструмента TextBox.

 – пиктограмма этого инструмента.

Оставляя небольшое пространство экранной формы для чертежа комнаты и для командной кнопки, размещаем 4 объекта Текстовое поле для входных данных и результата. Этот процесс аналогичен созданию объектов Метка.

Таким же способом помещаем на экранную форму еще два объекта: объект Командная кнопка и объект Изображение. Для этого используются инструменты CommandButton и Image.

 – пиктограмма инструмента CommandButton,

 – пиктограмма инструмента Image.

Окончательный вид экранной формы после размещения на ней всех объектов управления показан на рисунке Л1.4.

Последний из размещенных на форме объектов окружен восьмью маркерами. Это объект Изображение. Здесь будет находиться чертеж комнаты – прямоугольного параллелепипеда.

Наполним созданные объекты конкретным содержанием. Это содержание определяется значением их свойств.

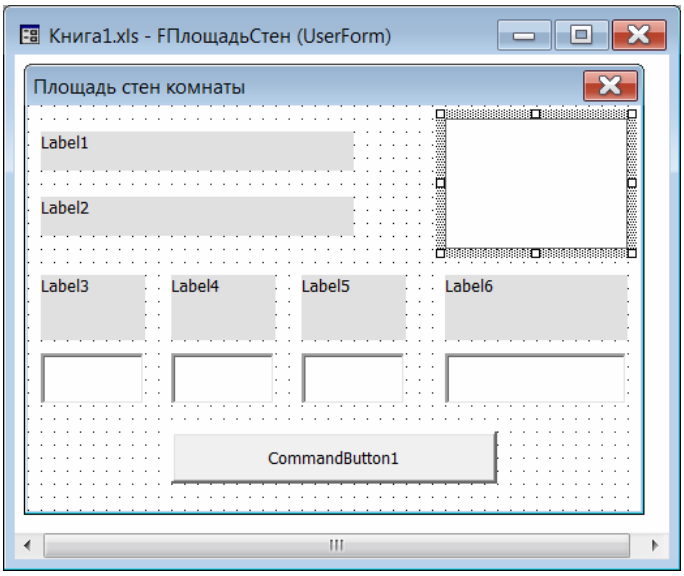


Рис. Л1.4. Окно с экранной формой приложения
«Площадь стен комнаты»

Обратите внимание, что в каждый момент проектирования только один объект является выделенным (активным). Как было только что отмечено (см. рис. Л1.4), на экранной форме он окружен рамкой из восьми маркеров. В окне свойств отображается список свойств именно активного объекта. При активизации другого объекта экранной формы изменяется содержимое «Окна свойств».

Объекты будут наполняться содержанием по порядку. Для этого их нужно поочередно активировать и работать с «Окном свойств» каждого активного объекта.

Определение свойств меток. Для созданных меток определим значения свойств `(Name)` и `Caption` (табл. Л1.1).

Таблица Л1.1

Имя метки по умолчанию	Значение свойства `(Name)`	Значение свойства `Caption`
`Label1`	`Метка1`	`Площадь стен комнаты вычисляется по формуле`
`Label2`	`Метка2`	`S=2*(A+B)*H`
`Label3`	`Метка3`	`Длина (A):`
`Label4`	`Метка4`	`Ширина (B):`
`Label5`	`Метка5`	`Высота (H):`
`Label6`	`Метка6`	`Площадь стен (S):`

Установим по своему усмотрению для созданных меток тип, размер и цвет шрифта (значения свойств `Font` и `ForeColor`). Для настройки

свойства Font нужно щелкнуть мышью в соответствующей строке окна свойств, а затем выполнить щелчок на кнопке ... – откроется окно диалога «Шрифт» (рис. Л1.5), в котором выбираются необходимые параметры шрифта. Выберем также цвет фона (значение свойства BackColor). Чтобы фон меток сливался с фоном формы, сделаем его белым.

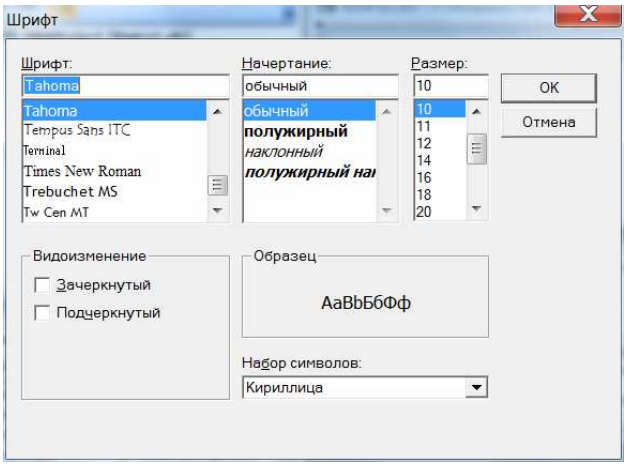


Рис. Л1.5. Окно диалога «Шрифт»

Определение свойств текстовых полей. Установим значения свойств четырех текстовых полей (объектов Текстовое поле).

Свойству (Name) текстовых полей присвоим следующие значения: *Длина*, *Ширина*, *Высота*, *Площадь*.

Установим начальные значения свойства Text всех четырех текстовых полей равными нулю.

Определение свойств командной кнопки. Свойству (Name) присвоим значение Командная Кнопка (без пробела), а свойству Caption – значение *Нажмите на эту кнопку, чтобы получить площадь стен*.

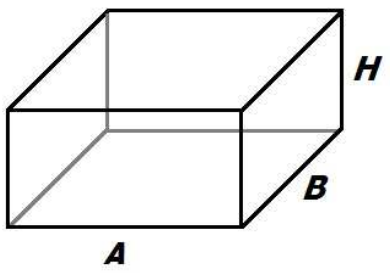


Рис. Л1.6. Чертеж комнаты для объекта Изображение

Определение свойств изображения. Для объекта Изображение установим значение свойства Picture.

Этим значением должен быть графический файл, находящийся на компьютере. Этот файл нужно предварительно создать, например, с помощью графического редактора *Paint* (рис. Л1.6.).

После установки значений свойств объектов экранная форма должна выглядеть как на рисунке Л1.7.

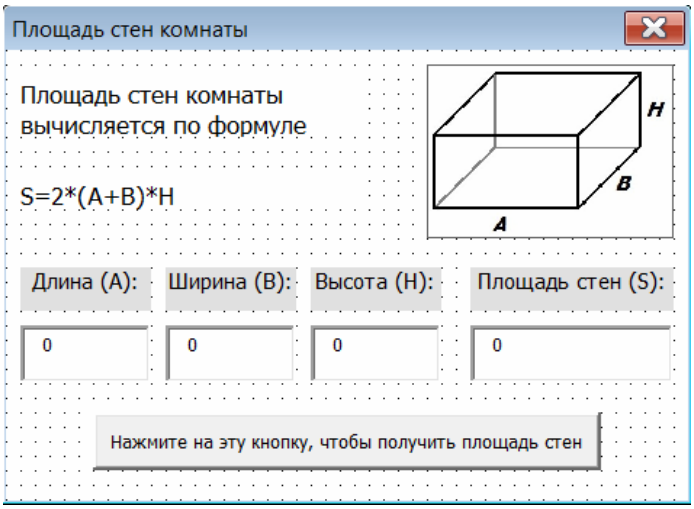


Рис. Л1.7. Вид экранной формы после установки значений свойств объектов

Программирование

Составить алгоритмы и написать программы – это главный этап проектирования приложения в среде Visual Basic. Но прежде чем приступить к этому этапу, необходимо определить те *события*, для которых нужно составить программный код.

В нашем примере в работе приложения будет использоваться только одно событие – *щелчок мышью командной кнопки*. Именно это событие будет будет запускать на выполнение программу вычисления площади стен.

Алгоритм решения задачи вычисления площади стен комнаты:

1) ввести три числа: `A`, `B` и `H` – длину, ширину и высоту;
2) найти площадь одной стены: `S1=A*H`;
3) найти площадь другой стены: `S2=B*H`;
4) удвоить сумму этих площадей: `S=2*(S1+S2);`
5) вывести результат: число `S` – площадь всех 4-х стен.

Для написания программного кода и привязки его к определенному событию необходимо открыть окно программного кода (рис. Л1.8). Сделать это можно следующим образом:

1) активизировать командную кнопку;
2) выполнить команду *меню View → Code*, или нажать функциональную клавишу F7 на клавиатуре, или выполнить двойной щелчок мышью на командной кнопке – в окне программного кода появится заготовка

для программного кода: первая и последняя строки процедуры для самого распространенного события Click, относящегося к объекту Командная кнопка.

Создаваемая процедура называется КомандныйКнопка_Click(). Первая строка программы начинается со слов Private Sub, а заканчивается программа словами End Sub.

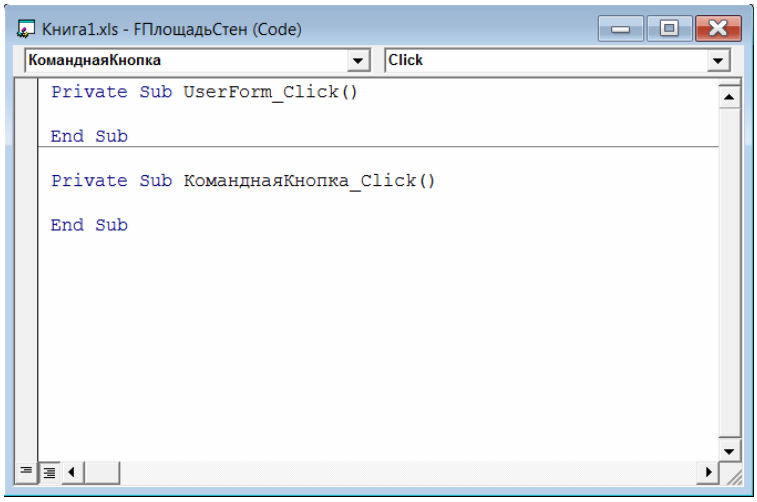


Рис. Л1.8. Окно программного кода командной кнопки

Между указанными двумя строками заготовки наберем с клавиатуры строки программного кода:

```
Private Sub КомандныйКнопка_Click()
   A = Val(Длина.Text)
   B = Val(Ширина.Text)
   H = Val(Высота.Text)
   S = 2 * (A + B) * H
   Площадь.Text = Str(S)
End Sub
```

Прокомментируем эту запись:

– последовательность строк кода (сверху вниз) соответствует последовательным шагам алгоритма решения данной задачи;

– знак = обозначает присваивание переменной определенного значения;

– знаки * и + обозначают операции умножения и сложения;

– выражение Длина.Text обозначает значение свойства Text объекта Длина (аналогично обозначаются значения этого свойства других объектов – Ширина, Высота и Площадь);

– запись `Val(X)` обозначает, что значение переменной `X` преобразуется из строки символов в число, а запись `Str(X)` – значение переменной `X` преобразуется из числа в строку символов.

Запуск программы

Запустить программу можно следующими способами:

– по команде *меню Run → Run Sub/UserForm*;

– по щелчку на кнопке «Run Sub/UserForm»;

– по нажатию функциональной клавиши F5 на клавиатуре.

На рисунке Л1.9 показано окно работающего приложения после того как были введены исходные данные, а затем выполнен щелчок мышью на командной кнопке, – в результате сработала процедура `КомандныяКнопка_Click()` и в поле `Площадь` появился результат.

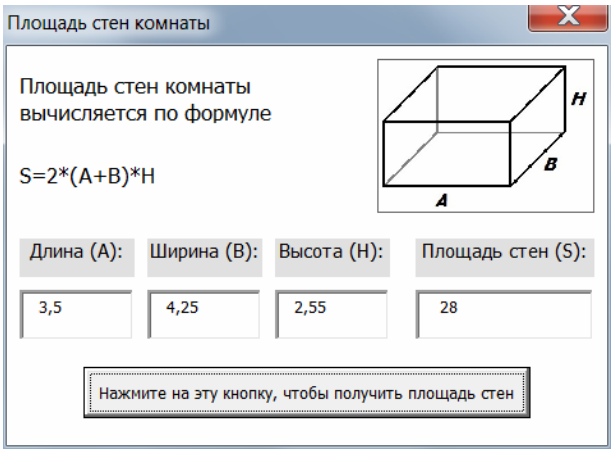


Рис. Л1.9. Окно работающего приложения

Завершение работы программы

Завершить работу программы (закрыть окно приложения) можно следующими способами:

– с помощью кнопки «Reset» на панели инструментов;

– щелчком по кнопке закрытия окна в правом верхнем углу окна приложения.

Отладка программы

Попытка запустить программу не всегда приводит к немедленному успеху. Очень часто (особенно когда программа не такая простая, как в рассмотренном примере) попытка запуска приводит к появлению сообще-

ний об ошибках. Для исправления ошибок в Visual Basic for Applications импользуется средство *Отладчик*.

Для вызова отладчика можно щелкнуть *правой* кнопкой мыши по панели инструментов и в открывшемся меню выбрать команду *Debug*. Выбор этой команды вызовет появление панели инструментов «Debug» – *Отладчика программ* (рис. Л1.10).

Рис. Л1.10. Панель инструментов «Debug»

Назначение кнопок этой панели инструментов приведено в таблице Л1.2.

Таблица Л1.2

Кнопка	Назначение кнопки
	Кнопка запуска программы (*Run*)
	Кнопка остановки программы (*Break*)
	Кнопка завершения работы программы (*End*)
	Отмечается строка текста, на которой необходимо прервать выполнение программы
	После остановки запускается один (очередной) оператор программы
	После остановки запускается и выполняется очередная процедура программы
	Открывается окно слежения (после остановки в это окно выводятся значения выбранных переменных)
	Прослеживаются значения выбранных переменных в окне слежения
	Открывается окно локальных переменных (после остановки в это окно выводятся значения переменных процедуры, в которой произошла остановка)
	Открывается окно немедленного выполнения (после остановки в нем можно записать и выполнить выражение на Visual Basic)

Для получения доступа к инструментам отладчика можно воспользоваться также *меню Debug*. При этом открывается меню с инструментами отладчика (рис. Л1.11).

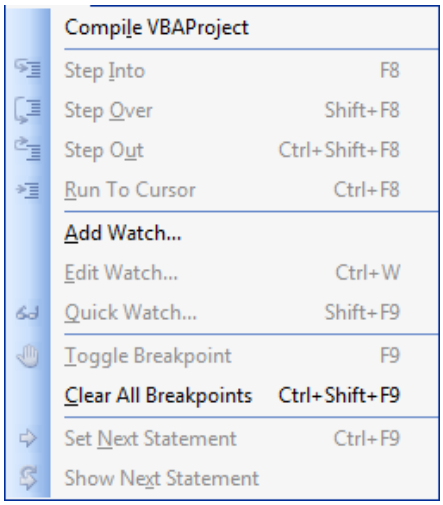


Рис. Л1.11. Меню «Debug»

Сохранение проекта

Прежде всего выберем проекту имя. По аналогии с именем формы (`FПлощадьСтен`) назовем проект `РПлощадьСтен`. Для этого в окне проводника проекта щелкнем правой кнопкой по имени проекта *VBAProject*. В появившемся контекстном меню выбрать команду *VBAProject Properties* – откроется окно диалога «VBAProject – Project Properties» (рис. Л1.12). В этом окне в поле «Project Name» ввести имя `РПлощадьСтен` и нажать кнопку OK.

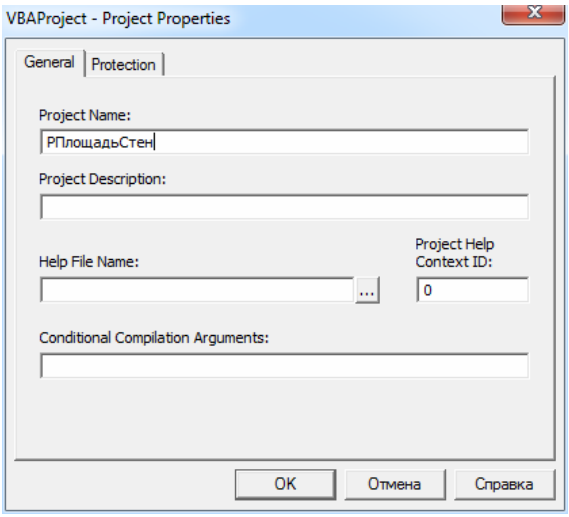


Рис. Л1.12. Окно диалога «VBAProject – Project Properties»

Сохранение рабочей книги

Сохранить рабочую книгу под именем «VBA_1_Фамилия» (без кавычек) в папке, указанной преподавателем.

ЛАБОРАТОРНАЯ РАБОТА № 2
Линейное программирование в VBA для Microsoft Excel

Цель работы: создание линейных программ в Excel VBA.
Теоретические сведения. См. разделы 3.2 и 4.1.1 – 4.1.8 лекций.

Постановка задачи № 1

Рассчитать прибыль фирмы по данным, приведенным в таблице Л2.1.

Таблица Л2.1

Выручка от реализации (ВР)	Себестоимость (С)	Внереализационный доход (ВД)	Балансовая прибыль (БП)	Налог на прибыль (НП)	Сумма налога (СН)	Размер прибыли (РП)
900,00 р.	400,00 р.	150,00 р.	= (ВР + ВД - С)	20,0 %	= (БП × НП)	= (БП - СН)

Решение

Создание экранной формы

Для создания пользовательской формы необходимо открыть новую рабочую книгу MS Excel и на листе `Лист1` в первой строке ввести наименование полей таблицы, а строкой ниже установить в ячейке, соответствующей графе НП, процентный формат, а в остальных шести – денежный (рис. Л2.1).

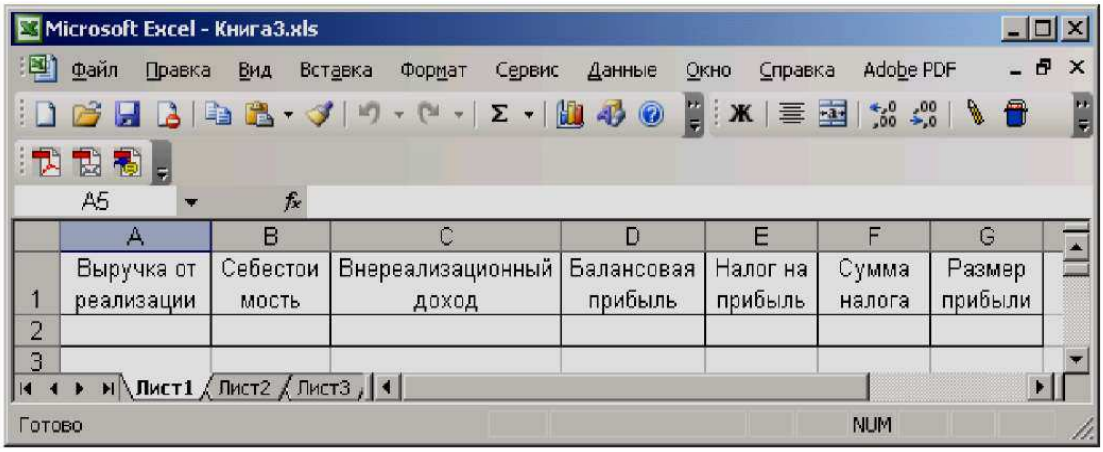


Рис. Л2.1. Шаблон таблицы «Расчет прибыли»

Сконструировать пользовательскую форму:

1) открыть среду проектирования VBA;

2) добавить новую форму, воспользовавшись командой *меню Insert → UserForm*;

3) разместить на форме с помощью панели элементов Toolbox следующие элементы управления:

– слева семь надписей (элемент Label);

– справа – семь полей (элемент TextBox);

– ниже – четыре командных кнопки (элемент CommandButton).

4) помощью окна свойств установить свойства размещенных объектов (табл. Л2.2). Пользовательская форма в окне редактора должна выглядеть как на рисунке Л2.2.

Таблица Л2.2

Объект	Свойство	Объект	Свойство
UserForm1	Caption = Расчет прибыли	TextBox6	Name = txtSN Locked = True
CommandButton 1	Name = calc Caption = Расчет	TextBox7	Name = txtRP Locked = True
CommandButton 2	Name = printToTable Caption = Заполнить таблицу	Labe1	Caption = Выручка от реализации AutoSize = True
CommandButton 3	Name = clean Caption = Очистить	Labe2	Caption = Себестоимость AutoSize = True
CommandButton 4	Name = exitForm Caption = Выход	Labe3	Caption = Внереализационный доход AutoSize = True
TextBox1	Name = txtVR Locked = False	Labe4	Caption = Налог на прибыль, % AutoSize = True
TextBox2	Name = txtS Locked = False	Labe5	Caption = Балансовая прибыль AutoSize = True
TextBox3	Name = txtVD Locked = False	Labe6	Caption = Сумма налога AutoSize = True
TextBox4	Name = txtNP Locked = False	Labe7	Caption = Размер прибыли AutoSize = True
TextBox5	Name = txtBP Locked = True		

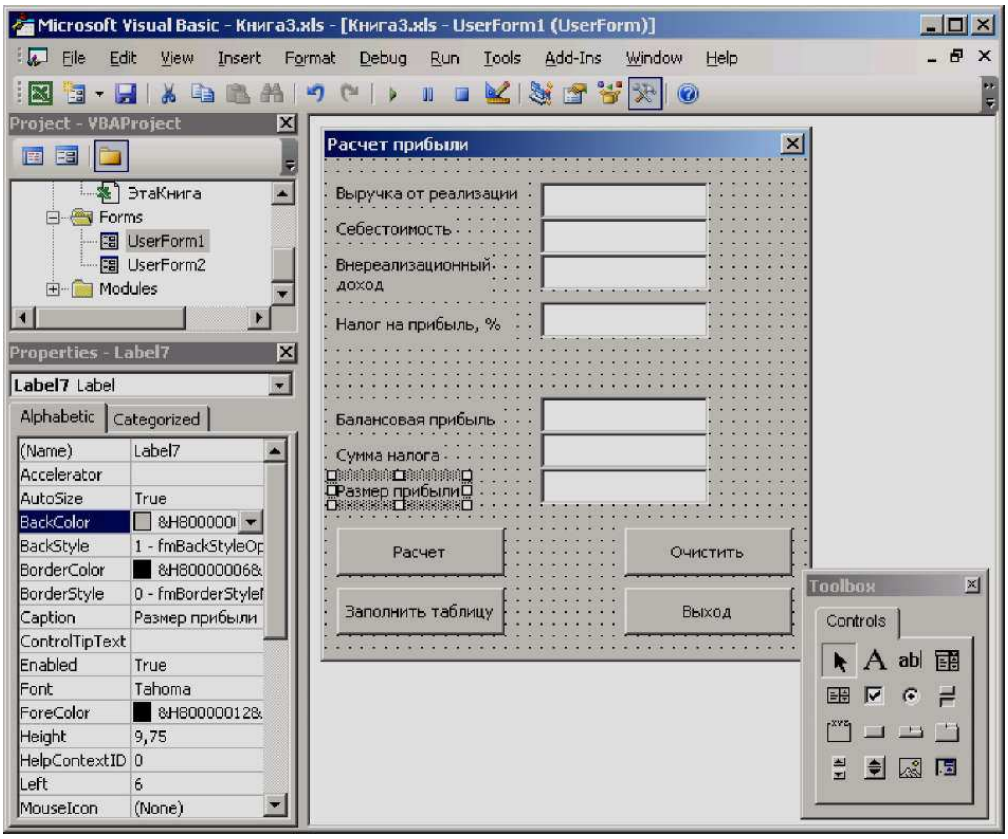


Рис. Л2.2. Форма «Расчет прибыли» в окне редактора VBA

Программирование

1) Создать процедуру обработки события `Click`, возникающего при нажатии на кнопку:

– выполнить двойной щелчок мыши по командной кнопке «Расчет» – откроется окно редактора кода VBA, в котором введем приведенный ниже программный код:

```
Dim VR, VD, S, NP As Single   'задание типа переменных
Dim BP, SN, RP As Single
Private Sub calc_Click()
    VR = Val(txtVR.Text) 'считывание значения выручки от реали-
    зации
    S = Val(txtS.Text)   'считывание значения себестоимости
    VD = Val(txtVD.Text) 'считывание значения внереализационно-
    го дохода
    NP = Val(txtNP.Text) / 100     'считывание значения налога
    на прибыль и перевод его в доли
    BP = VR + VD - S     'вычисление балансовой прибыли
    SN = BP * NP    'вычисление суммы налога
    RP = BP - SN    'вычисление размера прибыли
    txtBP.Text = BP 'вывод балансовой прибыли в текстовое поле
```

```
    txtSN.Text = SN 'вывод суммы налога в текстовое поле
    txtRP.Text = RP 'вывод размера прибыли в текстовое поле
    calc.BackColor = Rnd * 10^5 'изменение цвета фона кнопки
    для визуализации того, что процесс вычислений выполнен
End Sub
```

Примечание 2.1. Здесь функция *Val(строка)* преобразует строку в числовое выражение.

– закрыть окно программного кода;

2) аналогично ввести программный код для остальных кнопок:

```
Private Sub printToTable_Click()
   Cells(2, 1) = VR 'вывод значения выручки от реализации в
   ячейку A2
   Cells(2, 2) = S  'вывод значения себестоимости в ячейку B2
   Cells(2, 3) = VD 'вывод значения внереализационного дохода в
   ячейку C2
   Cells(2, 4) = BP 'вывод значения балансовой прибыли в ячейку
   D2
   Cells(2, 5) = NP  'вывод значения налога на прибыль в ячейку E2
   Cells(2, 6) = SN 'вывод значения суммы налога в ячейку F2
   Cells(2, 7) = RP  'вывод значения размера прибыли в ячейку G2
   printToTable.BackColor = Rnd * 10^5   'изменение цветового фо-
   на кнопки
End Sub

Private Sub clean_Click()
   txtVR.Text = Clear    'очистка текстовых полей
   txtS.Text = Clear
   txtVD.Text = Clear
   txtNP.Text = Clear
   txtBP.Text = Clear
   txtSN.Text = Clear
   txtRP.Text = Clear
   Cells(2, 1).ClearContents  'очистка ячеек A2:G2
   Cells(2, 2).ClearContents
   Cells(2, 3).ClearContents
   Cells(2, 4).ClearContents
   Cells(2, 5).ClearContents
   Cells(2, 6).ClearContents
   Cells(2, 7).ClearContents
End Sub

Private Sub exitForm_Click()
   End
   End Sub
```

Примечание 2.2. Свойство `Cells(i, j)` позволяет обращаться к содержимому ячейки, находящейся на пересечении строки с номером `i` и столбца с номером `j`.

3) вернуться на Лист1;

4) с помощью инструмента «Кнопка»  панели инструментов «Элементы управления» (*меню Вид → Панели инструментов → Элементы управления*) разместить ниже таблицы командную кнопку;

5) с помощью контекстного меню открыть окно свойств размещенной кнопки. В этом окне установить следующие значения свойств:

```
Caption = Форма для расчета прибыли,
Name = ОткрытьФорму;
```

6) двойным щелчком по кнопке перейти в редактор VBA и в модуле «Лист1» ввести следующий программный код:

```
Private Sub ОткрытьФорму_Click()
   UserForm1.Show   'вывод формы на экран
End Sub
```

7) перейти на Лист1 и проверить работу созданной формы:

– нажатием кнопки «Форма для расчета прибыли» открыть созданную форму;

– в поля «Выручка от реализации», «Себестоимость», «Внереализационный доход» и «Налог на прибыль,%» созданной формы ввести контрольные значения;

– нажатием кнопки «Расчет» рассчитать балансовую прибыль, сумму налога и размер прибыли;

– нажатием кнопки «Заполнить таблицу» перенести все значения из формы в диапазон ячеек A2:G2 рабочего листа Лист1. Форма и рабочий лист должны выглядеть как на рисунке Л2.3;

– нажатием кнопки «Очистить» удалить значения из всех полей формы и диапазона ячеек A2:G2 рабочего листа Лист1;

– нажатием кнопки «Выход» завершить работу формы.

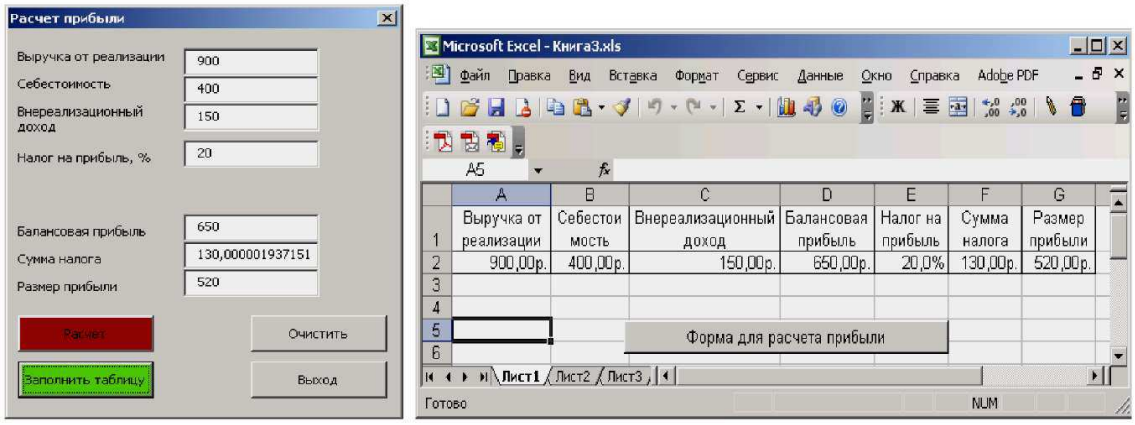


Рис. Л2.3. Форма и таблица «Расчет прибыли»

Сохранение рабочей книги

Сохранить рабочую книгу под именем «VBA_2_Фамилия» (без кавычек) в папке, указанной преподавателем.

Постановка задачи № 2

Используя Excel VBA, создать пользовательскую форму «Вычисление функций», предназначенную для вычисления выражений при заданных целых числах *x*, *y*, *z:*

$$a = \frac{1}{3}\sqrt{x} + \frac{1}{5}\sqrt[5]{y}; \qquad b = e^{2x+1}; \qquad c = \frac{2x^3 - 1}{tg^3 x - \sin y}; \qquad f = \sqrt{za} - \log_3 c + b.$$

Пользовательскую форму создать в рабочей книге *VBA_2_Фамилия*.

Создание экранной формы

1) открыть редактор VBA и добавить еще одну форму;

2) разместить на форме четыре метки и три командные кнопки, а с помощью окна свойств установить свойства этих объектов (табл. Л2.3). Внешний вид пользовательской формы приведен на рисунке Л2.4.

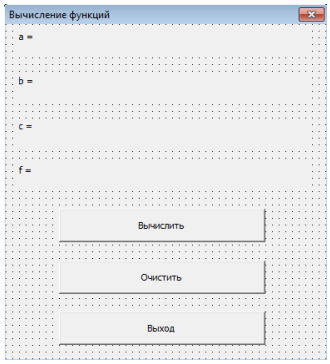


Рис. Л2.4. Форма «Вычисление функций»

Таблица Л2.3

Объект	Свойство	Объект	Свойство
`UserForm1`	`Caption = Вычисление функций`	`Label1`	`Caption = a = AutoSize = False`
`CommandButton1`	`Name = calc Caption = Вычислить`	`Label2`	`Caption = b = AutoSize = False`
`CommandButton2`	`Name = clean Caption = Очистить`	`Label3`	`Caption = c = AutoSize = False`
`CommandButton3`	`Name = exitForm Caption = Выход`	`Label4`	`Caption = f = AutoSize = False`

Программирование

1) Создать процедуры обработки события `Click`, возникающего при нажатии на кнопки:

```
Private Sub calc_Click()
   'определение типов переменных
   Dim x, y, z As Integer
   Dim a, b, c, f As Single
   'ввод значений аргументов функции
   x = Val(InputBox("Введите значение x", "Ввод данных"))
   y = Val(InputBox("Введите значение y", "Ввод данных"))
   z = Val (InputBox ("Введите значение z", "Ввод данных"))
   'вычисление функции
   a = Sqr(x) / 3 + y ^ (1 / 5) / 5
   'вывод значений функции
   MsgBox "При x = " & x & ", y = " & y & " функция a = " & a
   b = Exp(2 * x + 1)
   MsgBox "При x = " & x & " функция b = " & b
   c = (2 * x ^ 3 - 1) / (Tan(x) ^ 3 - Sin(y))
   MsgBox "При x = " & x & ", y = " & y & " функция c = " & c
   f = Sqr(z * a) - Log(c) / Log(3) + b
   MsgBox "При x = " & x & ", y = " & y & ", z = " & z &
   " функция f = " & f
   'вывод значений функции в надпись
   Label1.Caption = Label1.Caption + Str(a)
   Label2.Caption = Label2.Caption + Str(b)
   Label3.Caption = Label3.Caption + Str(c)
   Label4.Caption = Label4.Caption + Str(f)
End Sub

Private Sub clean_Click()
   Label1.Caption = "a = "
   Label2.Caption = "b = "
   Label3.Caption = "c = "
   Label4.Caption = "f = "
End Subъ

Private Sub exitForm_Click()
   End
End Sub
```

Примечание 2.3. Здесь функция `InputBox` используется для ввода информации в отдельном диалоговом окне и имеет следующий синтаксис (в квадратных скобках указаны необязательные параметры):

`InputBox («Текст сообщения», [«Текст заголовка диалогового окна»,] [значение текстового поля ввода по умолчанию])`

Примечание 2.4. Оператор `MsgBox` используется в качестве диалогового окна вывода сообщений и имеет следующий синтаксис:

`MsgBox «Текст сообщения», [buttons], [«Текст заголовка диалогового окна»],`

где `buttons` – числовое выражение, задающее параметры для кнопок управления и значков в диалоговом окне и состоящее из констант VBA.

Примечание 2.5. Оператор `&` предназначен для объединения символов в одну строку, а функция `Str(число)` возвращает текстовое представление исходного числа.

2) чтобы форма отображалась на экране при активизации рабочего листа, в редакторе VBA следует открыть модуль Лист2 и ввести следующий программный код:

```
Private Sub Worksheet_Activate()
   UserForm2.Show
End Sub
```

Теперь при переходе на лист Лист2, будет появляться форма «Вычисление функций»;

3) проверить работу созданной формы и программного кода при следующих значениях переменных: $x = 0$, $y = 1$, $z = 1$. При указанных значениях переменных должны быть получены следующие значения функций: $a = 0{,}2$; $b \approx 2{,}7183$; $c \approx 1{,}1884$; $f \approx 3{,}0084$ (рис. Л2.5);

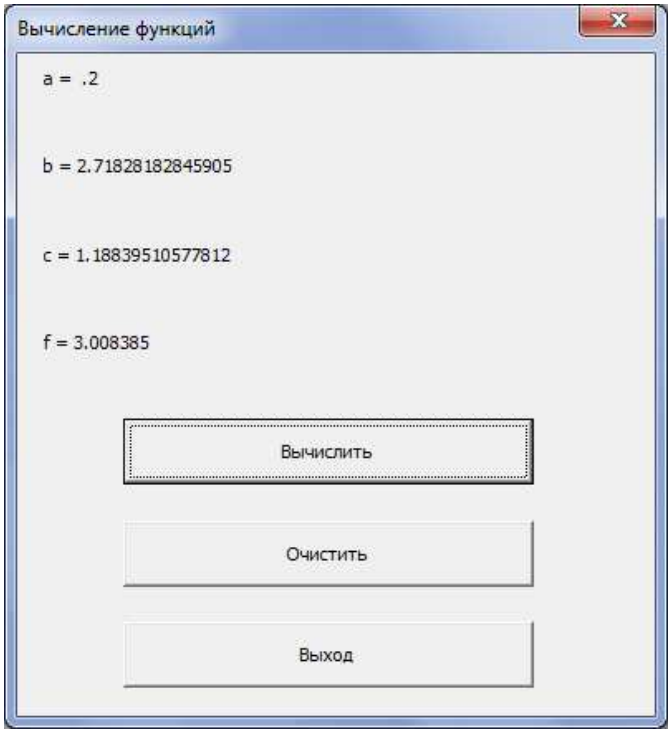


Рис. Л2.5. Результаты вычислений

4) переименовать Лист1 и Лист2 соответственно в Расчет прибыли и Вычисление выражений.

Сохранение рабочей книги

Сохранить рабочую книгу под текущим именем «VBA_2_Фамилия».

ЛАБОРАТОРНАЯ РАБОТА № 3
Программирование ветвлений в VBA для Microsoft Excel

Цель выполнения работы: создание программ ветвления в Excel VBA.
Теоретические сведения. См. раздел 4.1.14 лекций.

Постановка задачи № 1

Даны две окружности с радиусами R_1 и R_2 с центрами в точках $C_1\,(a_1, b_1)$, $C_2\,(a_2, b_2)$. Составить программу для определения характера расположения данных окружностей: пересекаются, касаются, не пересекаются.

Для решения поставленной задачи учесть, что если расстояние между центрами окружностей $\sqrt{(a_1-a_2)^2-(b_1-b_2)^2}$ больше суммы радиусов R_1 и R_2, то эти окружности не пересекаются; если равно сумме радиусов, то касаются; если меньше, то пересекаются.

Решение

1) Для решения задачи открыть новую рабочую книгу MS Excel и ввести на рабочий лист данные как на рисунке Л3.1.

	A	B	C	D	E
1	Окружность	Радиус	Координаты центра		Характер расположения
2			*a*	*b*	
3	Первая				
4	Вторая				

Рис. Л3.1. Таблица определения характера расположения окружностей

2) перейти в редактор VBA и в новом модуле (*меню Insert → Module, меню Insert → Procedure*) ввести следующий код:

```
Public Sub РасположениеОкружностей()
   Dim R1, R2, a1, a2, b1, b2 As Integer
     R1 = Range("B3").Value
     R2 = Range("B4").Value
     al = Range("C3").Value
     a2 = Range("C4").Value
     bl = Range("D3").Value
     b2 = Range("D4").Value
   'проверка условия
   If Sqr((a1 - a2) ^ 2 + (bl - b2) ^ 2) > R1 + R2 Then
      Range("E3:E4").Value = "окружности не пересекаются"
   'вывод результата
   ElseIf Sqr((a1 - a2) ^ 2 + (bl - b2) ^ 2) < R1 + R2 Then
      Range("E3:E4").Value = "окружности пересекаются"
   Else
      Range("E3:E4").Value = "окружности касаются"
   End If
End Sub
```

3) с помощью инструмента «Кнопка» панели инструментов «Формы» создать ниже таблицы кнопку для запуска процедуры `РасположениеОкружностей` (рис. Л3.2)

4) задав значения радиусов и координат центра окружностей, проверить работу программы.

Рис. Л3.2. Определение характера расположения окружностей

Сохранение рабочей книги

Сохранить рабочую книгу под именем «VBA_3_Фамилия» (без кавычек) в папке, указанной преподавателем.

Постановка задачи № 2

Брокер получает процент от суммы сделки. Для сделок, сумма которых менее 150 тыс. р., вознаграждение брокера составит 3,5%, от 150 до 500 тыс. р. – 4,5%, а для сделок, сумма которых составляет больше 500 тыс. р. – 5%. Разработать пользовательскую форму и составить программу для расчета вознаграждения брокера.

Решение

1) Создать пользовательскую форму как на рисунке Л3.3. Свойства объектов формы представлены в таблице Л3.1;

Таблица Л3.1

Объект	Свойство	Объект	Свойство
UserForm1	Caption = Вознаграждение брокера	Label1	Caption = Сумма сделки AutoSize = True
CommandButton1	Name = calc Caption = Рассчитать	Label2	Caption = Вознаграждение AutoSize = True
CommandButton2	Name = clean Caption = Очистить	TextBox1	Name = txtSumma Locked = True
CommandButton3	Name = exitForm Caption = Выход	TextBox2	Name = txtVzngr Locked = True

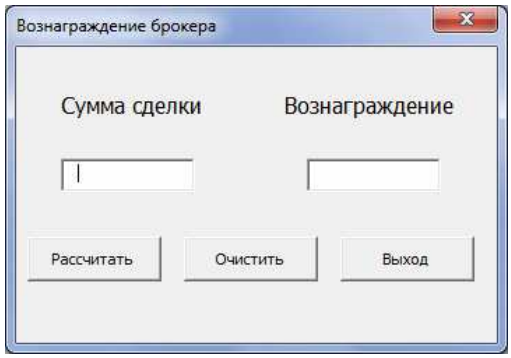


Рис. Л3.3. Форма «Вознаграждение брокера»

2) для обработки события `Click` кнопки «Рассчитать» ввести следующий код:

```
Private Sub calc_Click()
   Dim S, P As Single
   S = Val(InputBox("Введите сумму сделки", "Ввод данных"))
   If S < 150000 Then
     P = S * 0.035
   ElseIf (S >= 150000) And (S < 500000) Then
     P = S * 0.045
   Else
     P = S * 0.05
   End If
     txtSumma.Text = Str(S) + " руб."
     txtVzngr.Text = Str(P) + " руб."
End Sub
```

3) проверить работу программы.

Постановка задачи № 3

По введенной дате рождения сообщить пользователю, кто он по знаку зодиака (с 21 марта по 20 апреля – Овен, с 21 апреля по 20 мая – Телец, с 21 мая по 21 июня – Близнецы, с 22 июня по 22 июля – Рак, с 23 июля по 23 августа – Лев, с 24 августа по 22 сентября – Дева, с 23 сентября по 23 октября – Весы, с 24 октября по 22 ноября – Скорпион, с 23 ноября по 21 декабря – Стрелец, с 22 декабря по 20 января – Козерог, с 21 января по 19 февраля – Водолей, с 20 февраля по 20 марта – Рыбы).

Решение

1) Создать пользовательскую форму как на рисунке Л3.4. Свойства объектов формы представлены в таблице Л3.2;

Таблица Л3.2

Объект	Свойство	Объект	Свойство
UserForm1	Caption = Знак зодиака	Label1	Caption = Дата рождения AutoSize = True
CommandButton 1	Name = calc Caption = Определить знак	Label2	Caption = Знак зодиака AutoSize = True
CommandButton 2	Name = clean Caption = Очистить	TextBox1	Name = txtData Locked = False
CommandButton 3	Name = exitForm Caption = Выход	TextBox2	Name = txtZodiak Locked = True

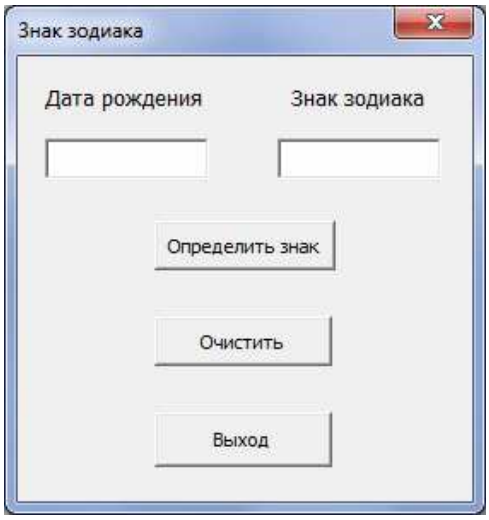


Рис. Л3.4. Форма «Знак зодиака»

2) для обработки события Click кнопки «Определить знак» написать программный код.

Чтобы определить знак зодиака по введенной пользователем дате рождения, воспользоваться оператором выбора Select Case. Кроме этого для выделения из даты месяца и числа потребуются функции Month(Дата) и Day(Дата), которые возвращают соответственно месяц как целое число от 1 до 12 и день как целое число от 1 до 31. Таким образом, код обработки события нажатия кнопки «Определить знак» будет иметь вид:

```
Private Sub calc_Click()
   Dim DR As Date
   DR = (txtData.Text)
      Select Case Month(DR) ‘выбор знака зодиака по номеру ме-
   сяца
   Case 1
      If Day(DR) <= 20 Then txtZodiak.Text = "Козерог" Else
```

```
      txtZodiak.Text = "Водолей"
   Case 2
      If Day(DR) <= 19 Then txtZodiak.Text = "Водолей" Else
      txtZodiak.Text = "Рыбы"
   Case 3
      If Day(DR) <= 20 Then txtZodiak.Text = "Рыбы" Else
      txtZodiak.Text = "Овен"
   Case 4
      If Day(DR) <= 20 Then txtZodiak.Text = "Овен" Else
      txtZodiak.Text = "Телец"
   Case 5
      If Day(DR) <= 20 Then txtZodiak.Text = "Телец" Else
   txtZodiak.Text = "Близнецы"
   Case 6
      If Day(DR) <= 21 Then txtZodiak.Text = "Близнецы" Else
      txtZodiak.Text = "Рак"
   Case 7
      If Day(DR) <= 22 Then txtZodiak.Text = "Рак" Else
      txtZodiak.Text = "Лев"
   Case 8
      If Day(DR) <= 23 Then txtZodiak.Text = "Лев" Else
      txtZodiak.Text = "Дева"
   Case 9
      If Day(DR) <= 22 Then txtZodiak.Text = "Дева" Else
      txtZodiak.Text = "Весы"
   Case 10
      If Day(DR) <= 23 Then txtZodiak.Text = "Весы" Else
      txtZodiak.Text = "Скорпион"
   Case 11
      If Day(DR) <= 22 Then txtZodiak.Text = "Скорпион" Else
      txtZodiak.Text = "Стрелец"
   Case Else
      If Day(DR) <= 21 Then txtZodiak.Text = "Стрелец" Else
      txtZodiak.Text = "Козерог"
   End Select
End Sub
```

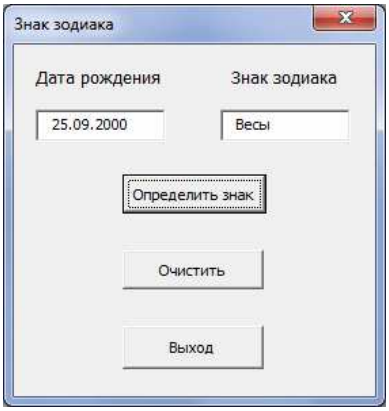


Рис. Л3.5. Форма «Определение знака зодиака»

3) проверить работу программы на следующем примере: родившиеся 25 сентября 2000 г. по знаку зодиака «Весы» (рис. Л3.5).

Сохранение рабочей книги

Сохранить рабочую книгу под текущим именем «VBA_3_Фамилия».

Постановка задачи № 4

Решить задачу № 2, используя оператор выбора.

Решение

1) В случае использования оператора выбора, код обработки события нажатия кнопки «Рассчитать» примет вид:

```
Private Sub calc Click()
   Dim S, P As Single
      S = Val(InputBox("Введите сумму сделки", "Ввод данных"))
      Select Case S
         Case 0 To 149999
            P = S * 0.035
         Case Is >= 500000
            P = S * 0.05
         Case Else
            P = S * 0.045
      End Select
      txtSumma.Text = Str(S) + " руб."
      txtVzngr.Text = Str(P) + " руб."
End Sub
```

Примечание 3.1. В данном случае слово `Is`, используемое в коде программы, является ключевым словом VBA, обозначающим тестируемое выражение в операторе `Case`.

Сохранение рабочей книги

Сохранить рабочую книгу под текущим именем «VBA_3_Фамилия».

ЛАБОРАТОРНАЯ РАБОТА № 4
Программирование циклических вычислительных процессов в VBA для Microsoft Excel

Цель выполнения работы: создание циклических программ в Excel VBA.

Теоретические сведения. См. раздел 4.1.14 лекций.

Постановка задачи № 1

Разработать пользовательскую форму и составить программу для определения индекса рентабельности предлагаемого инвестиционного проекта обновления оборудования, если сумма первоначальных вложений (ПВ) составляет 10 000 тыс. р., ставка дисконтирования r = 10%, а предполагаемые денежные потоки приведены в таблице Л4.1.

Таблица Л4.1

Год	0	1	2	3	4	5
Денежный поток (ДП), тыс. р.	-10 000	+5000	+4000	+3000	+2500	+1000

Решение

1) Для решения задачи открыть новую рабочую книгу MS Excel и ввести на рабочий лист данные как на рисунке Л4.1. Значения величин денежных потоков в 1-м –5-х годах будут считываться из ячеек рабочего листа «Пример 1»;

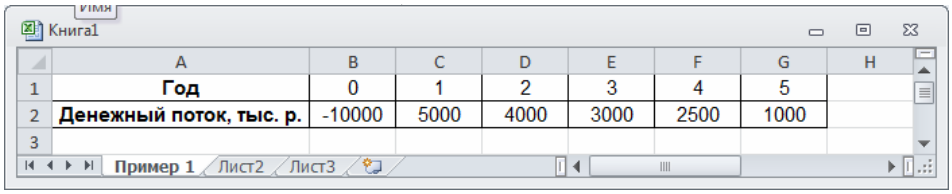


Рис. Л4.1. Таблица распределения денежных потоков

2) создать пользовательскую форму как на рисунке Л4.2. Свойства объектов формы представлены в таблице Л4.2.

Индекс рентабельности рассчитывается по формуле

$$ИР = \frac{\sum_{k=1}^{n} \frac{ДП_k}{(1+r)^k}}{ПВ},$$

где n – количество лет, в течение которых будут приходить денежные поступления;

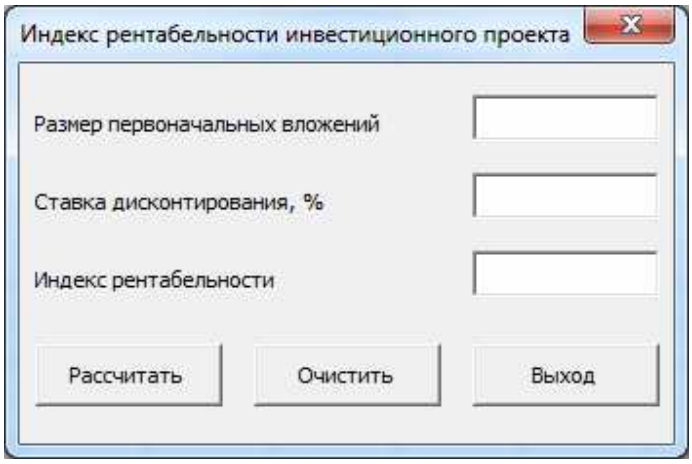


Рис. Л4.2. Форма «Индекс рентабельности инвестиционного проекта»

Таблица Л4.2

Объект	Свойство	Объект	Свойство
UserForm1	Caption = Индекс рентабельности инвестиционного проекта	Label1	Caption = Размер первоначальных вложений AutoSize = True
CommandButton1	Name = calc Caption = =Рассчитать	Label2	Caption = Ставка дисконтирования, % AutoSize = True
CommandButton2	Name = clean Caption = Очистить	Label3	Caption = Индекс рентабельности AutoSize = True
CommandButton3	Name = exitForm Caption = Выход	TextBox3	Name = txtIR Locked = True
TextBox1	Name = txtPV	TextBox2	Name = txtSD

3) для обработки события Click кнопки «Рассчитать» ввести приведенный ниже код;

```
Private Sub calc_Click()
   Dim pv, dp, i, n As Integer     'задание типа переменных
   Dim r, s As Single
      pv = Val(txtPV.Text)    'считывание значений переменных
      r = Val(txtSD.Text) / 100
      n = Val(InputBox("Введите количество лет, в течение кото-
   рых будут приходить денежные поступления ", "Ввод данных"))
   s = 0
   For i = 1 To n
       dp = Cells(2, i + 2)   'считывание значений денежных
                              'потоков
       s = s + dp / (1 + r) ^ i 'суммирование
   Next i
```

```
   txtlR.Text = Round(s / pv, 2) 'округление и вывод результата
End Sub
```

Примечание 4.1. Функция `Round(N, k)` округляет число `N` до `k` знаков после запятой.

4) с помощью инструмента «Кнопка» панели инструментов «Элементы управления» разместить ниже таблицы данных кнопку «Открыть форму» для запуска процедуры «ФормаИндексРентабельности» (рис. Л4.3). Код данной процедуры приведен ниже:

```
Private Sub ФормаИндексРентабельности_Click ()
   UserForm1.Show
End Sub
```

5) с помощью созданной кнопки «Открыть форму» запустить форму и рассчитать индекс рентабельности инвестиционного проекта (рис. Л4.3).

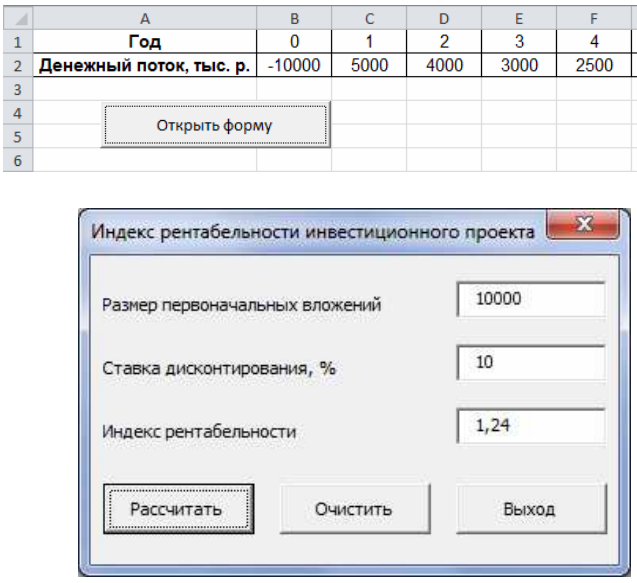


Рис. Л4.3. Определение индекса рентабельности инвестиционного проекта

Сохранение рабочей книги

Сохранить рабочую книгу под именем «VBA_4_Фамилия» (без кавычек) в папке, указанной преподавателем.

Постановка задачи № 2

Разработать пользовательскую форму и составить программу для поиска всех двузначных чисел, сумма квадратов цифр которых делится на 13.

Решение

1) Для решения задачи перейти в редактор VBA (в той же рабочей книге, что и в задаче № 1) и добавить еще одну форму как на рисунке Л4.4. Свойства объектов формы представлены в таблице Л4.3;

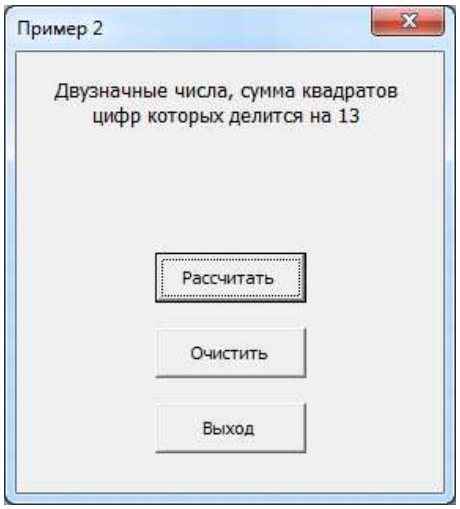


Рис. Л4.4. Форма для задачи 2

Таблица Л4.3

Объект	Свойство	Объект	Свойство
UserForm2	Caption = Пример_2	Command-Button1	Name = calc Caption = Рассчитать
Labell	Caption = Двузначные числа, сумма квадратов цифр которых делится на 13 AutoSize = True TextAlign = =2-fmTextAlignCenter	Command-Button2	Name = clean Caption = Очистить
Label2	Caption = " " AutoSize = False	Command-Button3	Name = exitForm Caption = Выход

2) для обработки событий Click кнопок «Рассчитать», «Очистить» и «Выход» написать приведенный ниже программный код:

```
Private Sub calc_Click()
   Label2.Caption = ""
   For i = 11 To 99
      i1 = i \ 10   'определение первой цифры двузначного числа
      i2 = i Mod 10 'определение второй цифры двузначного числа
      If (i1^2+i2^2) Mod 13=0 Then Label2.Caption =
      =Label2.Caption+Str(i)+" "
```

```
   Next i
End Sub
Private Sub clean_Click()
   Label2.Caption = ""
End Sub
Private Sub exitForm_Click()
   End
End Sub
```

3) на рабочем листе «Пример 2» разместить кнопку «Открыть форму» для вызова формы «Пример 2» и запустить данную форму (рис. Л4.5).

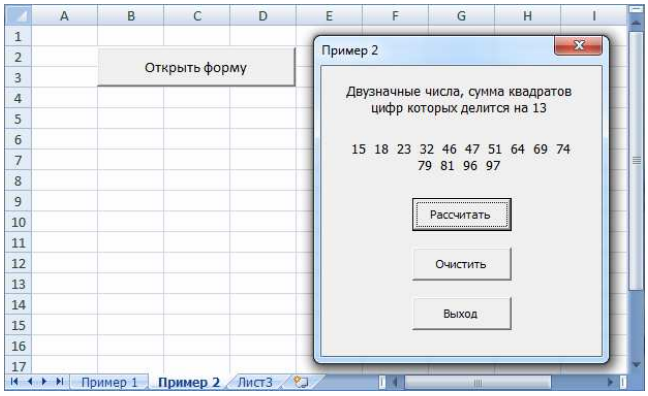


Рис. Л4.5. Двузначные числа, сумма квадратов цифр которых делится на 13

Сохранение рабочей книги

Сохранить рабочую книгу под текущим именем «VBA_5_Фамилия».

Постановка задачи № 3

Разработать пользовательский интерфейс и составить программу для вычисления с точностью $\varepsilon = 10^{-4}$ суммы ряда

$$S = \sum_{k=1}^{\infty} \frac{(-1)^{k+2}}{k!2^k}.$$

Задачу решить в той же рабочей книге Excel, что и задачи № 1 и № 2.

Решение

1) С помощью инструментов «Надпись», «Поле» и «Кнопка» панели инструментов «Элементы управления» разместить на рабочем листе «Пример 3» соответствующие объекты (рис. Л4.6). Свойства объектов листа «Пример 3» приведены в таблице Л4.4;

Таблица Л4.4

Объект	Свойство	Объект	Свойство
Label1	Caption = Сумма ряда AutoSize = True	Command-Button1	Name = calc Caption = Рассчитать
TextBox1	Name = txtS Locked = True	Command-Button2	Name = clean Caption = Очистить

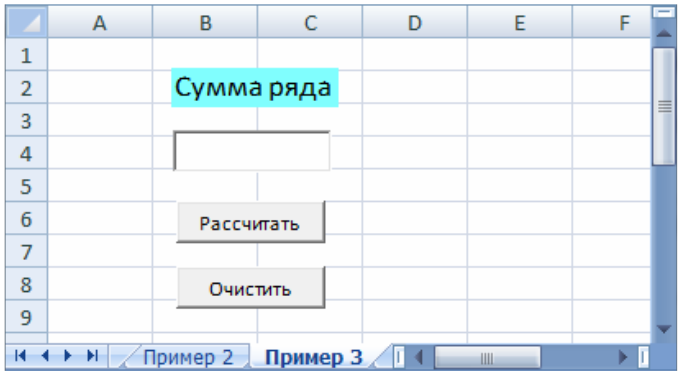


Рис. Л4.6. Интерфейс для вычисления суммы ряда

2) для обработки событий `Click` кнопок «Рассчитать» и «Очистить» написать в модуле «Лист3(Пример 3)» приведенный ниже программный код;

```
Private Sub calc_Click()
   Dim eps, a, s As Single
   Dim k, f As Integer
      eps = Val(InputBox("Задайте точность", "Ввод данных"))
         k = 1 'задание начальных данных
         f = k 'f соответствует факториалу числа k
         a = (-1)^(k + 2)/(f * 2^k)     'a соответствует от-
         дельному слагаемому ряда
         s = a 's соответствует сумме ряда
      Do While Abs(a) > eps
         k = k + 1
         f = f * k
         a = (-1) ^ (k + 2) / (f * 2 ^ k)
         s = s + a
      Loop
      txtS.Text = s
End Sub
Private Sub clean_Clicki
   txtS.Text = ""
End Sub
```

Примечание 4.2. Здесь тело цикла `Do While – Loop` будет выполняться, пока слагаемое ряда по абсолютной величине (т. к. $\left|\sum_{k=1}^{n}\frac{(-1)^{k+2}}{k!2^k}-\sum_{k=1}^{n-1}\frac{(-1)^{k+2}}{k!2^k}\right|=\left|\frac{(-1)^{n+2}}{n!2^n}\right|$) будет превышать заданную точность 10^{-4}.

3) проверить работу программы (рис. Л4.7).

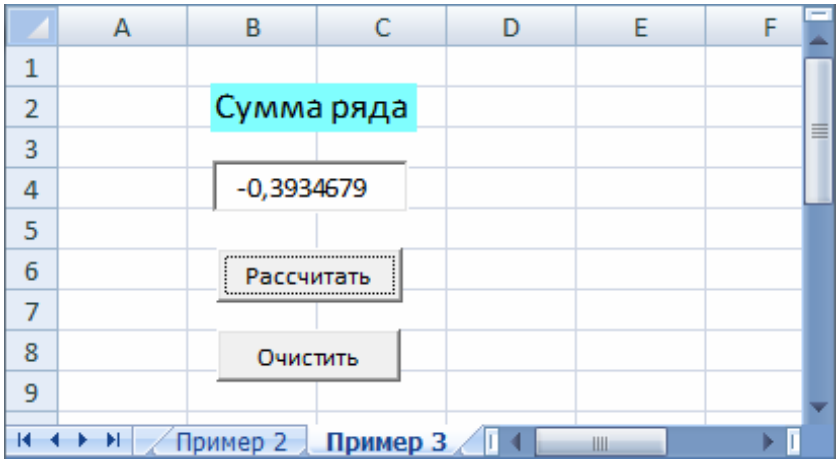


Рис. Л4.7. Результат вычисления суммы ряда

Сохранение рабочей книги

Сохранить рабочую книгу под текущим именем «VBA_6_Фамилия».

Постановка задачи № 4

Разработать пользовательскую форму и составить программу для решения следующей задачи.

При анализе хозяйственной деятельности предприятия была установлена следующая закономерность:

– в январе 2004 г. прибыль данного предприятия составила 573 тыс. р. и в течение следующих 5 лет ежемесячно увеличивалась на i %, где i – номер месяца (т е. в феврале – на 2%, в марте – на 3% и т. д.), по сравнению с суммой прибыли в предыдущем месяце.

Определить полученные предприятием суммы прибыли за 2004 г., 2005 г., ..., 2008 г., а также общую сумму прибыли за эти 5 лет.

Решение

1) Для решения задачи перейти в редактор VBA (в той же рабочей книге, что и в задачах № 1–3) и добавить еще одну форму как на рисунке Л4.8. Вывод вычисленных значений прибыли предприятия будет осуществляться как в поле объекта список (`ListBox`), размещенного на форме, так и в ячейки рабочего листа «Пример 4» (рис. Л4.9). Свойства объектов формы представлены в таблице 4.5;

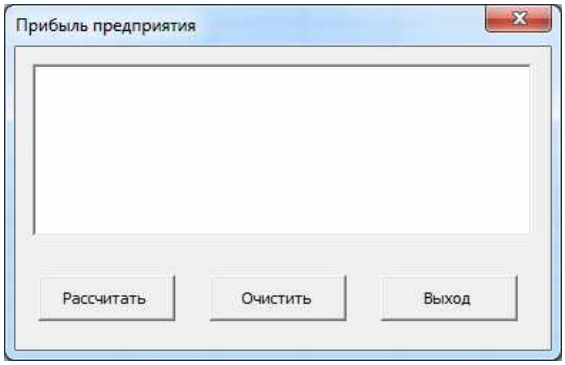


Рис. Л4.8. Форма для задачи № 4

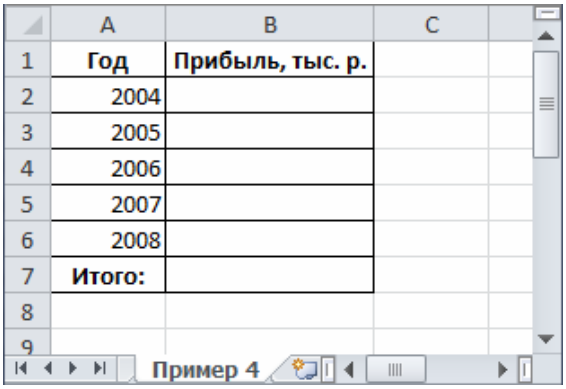


Рис. Л4.9. Таблица к задаче № 4

Таблица Л4.5

Объект	Свойство	Объект	Свойство
CommandButton1	Name = calc Caption = Рассчитать	UserForm3	Caption = Прибыль предприятия
CommandButton2	Name = clean Caption = Очистить	ListBox1	Name = ListPr ColumnCount = 2
CommandButton3	Name = exitForm Caption = Выход		

Примечание 4.3. Свойство `ColumnCount` задает количество колонок объекта `ListBox`.

2) для обработки событий `Click` кнопок «Рассчитать», «Очистить» и «Выход» написать приведенный ниже программный код;

```
Private Sub calc_Click()
   Dim i, j As Integer
   Dim PrM, prG, sPr As Single
         PrM = Val(InputBox("Введите прибыль, полученную в январе
      2004 г.", "Ввод данных"))
```

```
        ListPr.AddItem "Год", 0 'задание начальных подписей и
                                'данных
        ListPr.List(0, 1) = "Прибыль, тыс. р."
    sPr = 0
    For i = 4 To 8   'цикл по годам
       prG = 0
       For j = 1 To 12    'цикл по месяцам каждого года
          If (i = 4) And (j = 1) Then
             PrM = PrM
          Else
             PrM = PrM * (1 + j / 100)
          End If
       prG = prG + PrM
       Next j
       ListPr.AddItem "200" + Str(i), i- 3    'вывод года и
                                             'прибыли в объект ListBox
       ListPr.List(i - 3, 1) = Str(Round(prG, 3))
       Cells(i - 2, 2) = Round(prG, 3)   'вывод прибыли в ячейки
                                         'рабочего листа
       sPr = sPr + prG
    Next i
    ListPr.AddItem "Общая прибыль", 6
    ListPr.List(6,1) = Str (Round (sPr, 3))     'вывод общей прибыли
                                                'в объект ListBox
    Cells(i - 2, 2) = Round(sPr, 3) 'вывод общей прибыли в ячейки
                                    'листа
End Sub

Private Sub clean_Click()
   ListPr.Clear
End Sub

Private Sub exitForm_Click()
   End
End Sub
```

Примечание 4.4. В данном программном коде для расчета прибыли за каждый год и за 5 лет в целом используются вложенные циклы с параметром.

Примечание 4.5. Для вывода результатов вычислений в поле объекта `ListBox` используется метод `Addltem`, предназначенный для добавления элементов в список:

`ИмяСписка.AddItem Выражение, Индекс`, где

`Выражение` – элемент списка, который надо добавить;

`Индекс` – порядковый номер элемента в списке (нумерация элементов списка начинается с `0`).

Примечание 4.6. Свойство `List(НомерСтроки, НомерСтолбца)` объекта `ListBox` возвращает (присваивает) элемент списка, стоящий на пересечении указанных строки и столбца.

3) на рабочем листе «Пример 4» разместить кнопку «Открыть форму» для вызова формы «Прибыль предприятия» и запустить данную форму (рис. Л4.10);

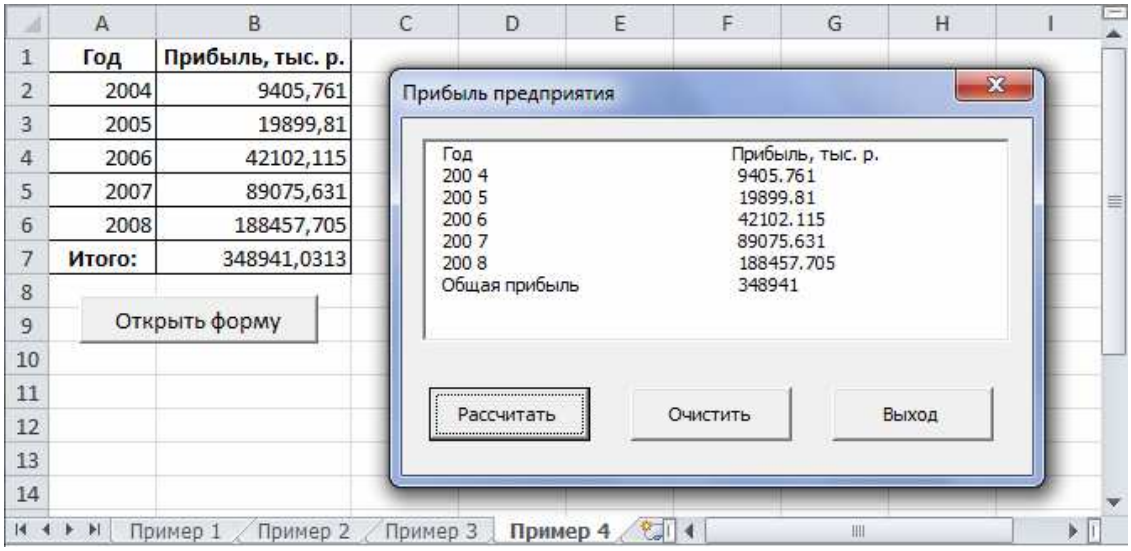


Рис. Л4.10. Результат вычисления прибыли предприятия

4) создать рабочий лист `Оглавление`:

– добавить в рабочую книгу MS Excel новый лист,

– добавленному листу присвоить имя `Оглавление`,

– в редакторе VBA добавить новый модуль по команде *меню Insert → Module*, в котором ввести следующий код:

```
Public Sub ЛистПример1()
   Worksheets("Пример 1").Activate
End Sub
Public Sub ЛистПример2()
   Worksheets("Пример 2").Activate
End Sub
Public Sub ЛистПример3()
   Worksheets("Пример 3").Activate
End Sub
Public Sub ЛистПример4()
   Worksheets("Пример 4").Activate
End Sub
```

– на листе `Оглавление` с помощью инструмента «Кнопка» панели инструментов «Формы» создать четыре кнопки: «Индекс рентабельности», «Двузначные числа», «Сумма ряда», «Прибыль предприятия»,

– назначить созданным кнопкам соответствующие макросы. Теперь при нажатии на каждую из этих кнопок будет осуществляться переход на лист с определенным заданием (рис. Л4.11),

– проверить работу созданных кнопок.

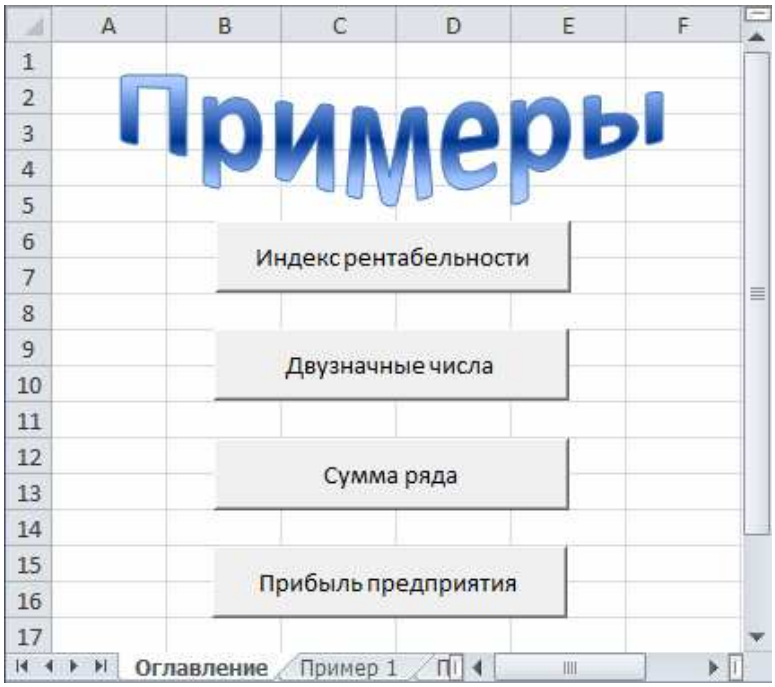


Рис. Л4.11. Лист «Оглавление»

Сохранение рабочей книги

Сохранить рабочую книгу под текущим именем «VBA_7_Фамилия».

ЛАБОРАТОРНАЯ РАБОТА № 5
Одномерные массивы в VBA для Microsoft Excel

Цель выполнения работы: изучение приемов работы с одномерными массивами в Excel VBA.

Теоретические сведения. См. разделы 4.1.9 и 4.1.14 лекций.

Постановка задачи

Известны данные о населении 12 государств: *Страна*, *Численность населения* (в миллионах жителей). Необходимо разработать пользовательскую форму и составить программу в редакторе Excel VBA, с помощью которой будут осуществляться следующие действия:

– ввод исходных данных и их вывод в виде таблицы на рабочий лист;

– определение государства, в котором проживает больше всего жителей, а также количество и название стран, в которых численность населения составляет от 10 до 50 миллионов жителей;

– вставка записи о численности населения еще одного государства, которая должна быть расположена после записи, соответствующей государству с наибольшим числом населения;

– сортировка полученных данных по полю *Страна* в алфавитном порядке.

Значения массива, соответствующего названиям стран, будут считываться с рабочего листа MS Excel. А значения массива, отвечающего за численность населения, будут вводиться с помощью функции `InputBox`.

Решение

1) Для решения задачи открыть новую рабочую книгу MS Excel и ввести названия стран в ячейки первого листа (рис. Л5.1);

	A	B
1	Страна	Численность населения, млн. чел.
2	Великобритания	
3	Франция	
4	Германия	
5	Испания	
6	Греция	
7	Италия	
8	Австрия	
9	Швеция	
10	Турция	
11	Швейцария	
12	Польша	
13	Португалия	

Рис. Л5.1. Таблица «Численность населения европейских государств»

2) в редакторе VBA создать пользовательскую форму как на рисунке Л5.2. Свойства объектов формы представлены в таблице Л5.1;

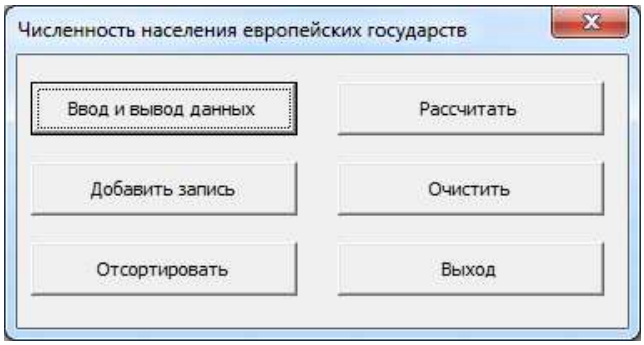


Рис. Л5.2. Форма «Численность населения европейских государств»

Таблица Л5.1

Объект	Свойство	Объект	Свойство
CommandButton1	Name = InputOutputData Caption = Ввод и вывод данных	CommandButton4	Name = calc Caption = Рассчитать
CommandButton2	Name = add Caption = Добавить запись	CommandButton5	Name = clean Caption = Очистить
CommandButton3	Name = sort Caption = Отсортировать	CommandButton6	Name = exitForm Caption = Выход
UserForm1	Caption = Численность населения европейских государств		

3) для обработки события Click кнопки «Ввод и вывод данных» написать приведенный ниже программный код:

```
Dim NazvGos(1 To 13) As String        'объявление типа переменных
Dim KolNasel(1 To 13) As Single
Dim SizeM, NomerMax, i, k As Integer

Private Sub InputOutputData_Click()
   SizeM = 12
   For i = 1 To SizeM
'ввод элементов массивов
     NazvGos(i) = Worksheets("Лист1").Range("A" & i + 1).Value
     KolNasel(i) = Val(InputBox("Введите численность населения
     страны " & NazvGos(i) & " в млн. чел.", "Ввод данных"))
     Cells(i + 1, 2) = KolNasel(i)   'вывод элементов массива
   Next i
   For i = 1 To SizeM + 1     'задание формата ячеек таблицы
      If i = 1 Then
```

```
'начертание шрифта
         Worksheets("Лист1").Range("A" & i).Font.FontStyle =
         = "полужирный"
         Worksheets("Лист1").Range("B" & i).Font.FontStyle =
         = "полужирный"
'горизонтальное выравнивание
         Worksheets("Лист1").Range("A" &
         i).HorizontalAlignment = xlCenter
         Worksheets("Лист1").Range("B" & i). HorizontalAlign-
         ment = xlCenter
      End If
'задание стиля границ ячеек
         Worksheets("Лист1").Range("A" & i).Borders.LineStyle
         = xlContinuous
         Worksheets("Лист1").Range("B" & i).Borders.LineStyle
         = xlContinuous
'числовой формат ячейки
         If i>1 Then
          Worksheets("Лист1").Range("B" & i).NumberFormat
         = "0.000"
      Next i
End Sub
```

4) при нажатии на кнопку «Рассчитать» на первом листе ниже таблицы будет выводиться название государства, в котором проживает больше всего жителей, а также количество и название стран, в которых численность населения составляет от 10 до 50 млн. жителей. Для обработки события нажатия кнопки «Рассчитать» написать приведенный ниже программный код;

```
Private Sub calc_Click()
   Dim maxKN As Single
     maxKN = KolNasel(1)
     k = 0
     Cells(SizeM + 3, 1) = "Наиболее населенная страна"
     Cells(SizeM + 4, 1) = "Число стран с населением 10-50
                           млн.чел."
     Cells(SizeM + 5, 1) = "  а именно:"
     For i = 1 To SizeM
   'поиск максимального элемента и его номера по массиву «Чис-
   ленность населения»
      If KolNasel(i) > maxKN Then maxKN = KolNasel(i): NomerMax =
     i
   'определение количества и названий стран с населением от 10
   до 50 млн.чел.
      If (KolNasel(i) > 10) And (KolNasel(i) < 50) Then
         k = k + 1
         Cells(SizeM + 4 + k, 2) = NazvGos(i)
```

```
      End If
     Next i
     Cells(SizeM + 3, 2) = NazvGos(NomerMax)
     Cells(SizeM + 4, 2) = Str(k)
End Sub
```

5) при нажатии на кнопку «Добавить запись» на втором листе будет выводиться таблица, содержащая информацию о численности населения 13 стран, т.е. после записи, соответствующей государству с наибольшим числом населения, еще будет добавлена следующая запись «Болгария 10,392». Для обработки события `Click` кнопки «Добавить запись» написать приведенный ниже программный код;

```
Private Sub add_Click()
   For i = SizeM + 1 To NomerMax + 2 Step -1 'раздвигаем элементы
   массива
      NazvGos(i) = NazvGos(i - 1)
      KolNasel(i) = KolNasel(i - 1)
   Next i
   NazvGos(NomerMax + 1) = "Болгария"   'вставляем элемент
   KolNasel(NomerMax + 1) = 10.392
   Worksheets("Лист2").Select           'переход на «Лист2»
      Cells(1, 1) = "Страна"
      Cells(1, 2) = "Численность населения (млн.чел.)"
   For i = 1 To SizeM + 1
      Cells(i + 1, 1) = NazvGos(i) 'вывод массивов на новый
   лист
      Cells(i + 1, 2) = KolNasel(i)
      Next i
   For i = 1 To SizeM + 2     'задание формата ячеек таблицы
      If i = 1 Then
         Worksheets("Лист2").Range("A" & i).Font.FontStyle =
         "полужирный"
         Worksheets("Лист2").Range("A" &
         i).HorizontalAlignment = xlCenter
         Worksheets("Лист2").Range("B" & i).Font.FontStyle =
         "полужирный"
         Worksheets("Лист2").Range("B" &
         i).HorizontalAlignment = xlCenter
      End If
         Worksheets("Лист2").Range("A" & i).Borders.LineStyle =
         xlContinuous
         Worksheets("Лист2").Range("B" & i).Borders.LineStyle =
         xlContinuous
      If i>1 Then Worksheets("Лист2").Range("B" &
         i).NumberFormat = "0.000"
      Next i
End Sub
```

6) при нажатии на кнопку «Отсортировать» будет выполняться сортировка массивов по названиям стран в алфавитном порядке, а результат сортировки будет выведен на третий лист. Для обработки события `Click` кнопки «Отсортировать» написать приведенный ниже программный код;

```
Private Sub sort_Click()
   Dim j As Integer 'объявление типа переменных
   Dim TempG As String
   Dim TempN As Single
      Worksheets("Лист3").Select
      Cells(1, 1) = "Страна"
      Cells(1, 2) = "Численность населения (млн.чел.) "
      For i = 1 To SizeM 'сортировка массива по возрастанию
         For j = i + 1 To SizeM + 1
            If NazvGos(i) > NazvGos(j) Then
               TempG = NazvGos(i):      TempN = KolNasel(i)
               NazvGos(i) = NazvGos(j): KolNasel(i) =
               KolNasel(j)
               NazvGos(j) = TempG:      KolNasel(j) = TempN
            End If
         Next j
      Next i
      For i = 1 To SizeM + 1
         Cells(i + 1, 1) = NazvGos(i)  'вывод массивов на но-
                                    вый лист
         Cells(i + 1, 2) = KolNasel(i)
      Next i
      For i = 1 To SizeM + 2  'задание формата ячеек таблицы
         If i = 1 Then
            Worksheets("Лист3").Range("A" & i).Font.FontStyle =
            "полужирный"
            Worksheets("Лист3").Range("A" &
            i).HorizontalAlignment = xlCenter
            Worksheets("Лист3").Range("B" & i).Font.FontStyle =
            "полужирный"
            Worksheets("Лист3").Range("B" &
            i).HorizontalAlignment = xlCenter
         End If
         Worksheets("Лист3").Range("A" & i).Borders.LineStyle =
         xlContinuous
         Worksheets("Лист3").Range("B" & i).Borders.LineStyle =
         xlContinuous
         If i>1 Then Worksheets("Лист3").Range("B" &
         i).NumberFormat = "0.000"
      Next i
End Sub
```

7) на первом рабочем листе поместить кнопку «Форма» для вызова формы «Численность населения европейских государств»;

8) запустить форму «Численность населения европейских государств» и проверить работу составленной программы (рис. Л5.3 и Л5.4).

Примечание 5.1. При вводе числовых данных целую и дробную часть отделять друг от друга точкой!

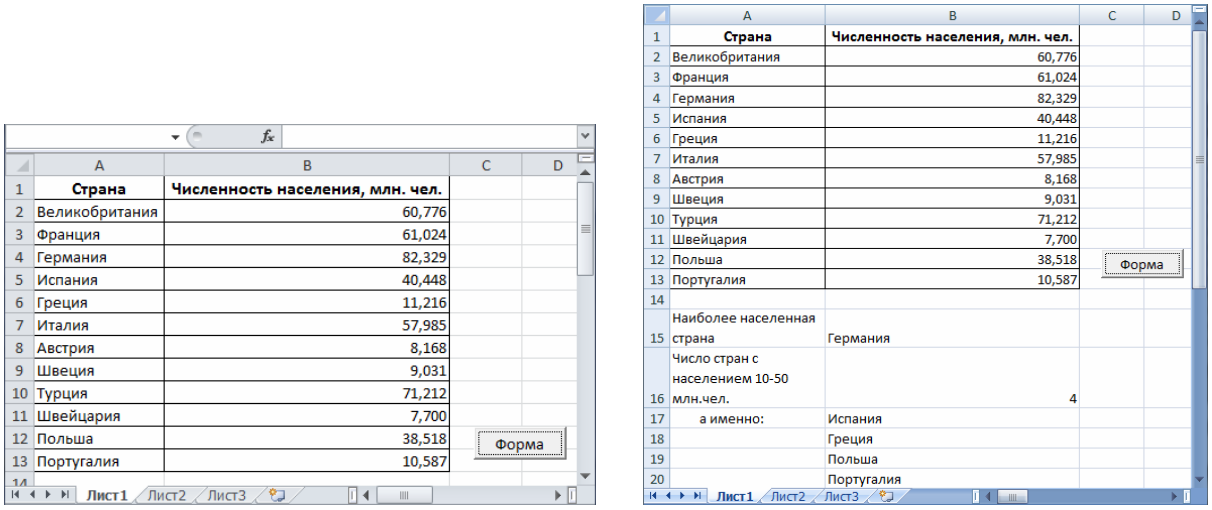


Рис. Л5.3. Результат нажатия кнопок «Ввод и вывод данных» и «Рассчитать»

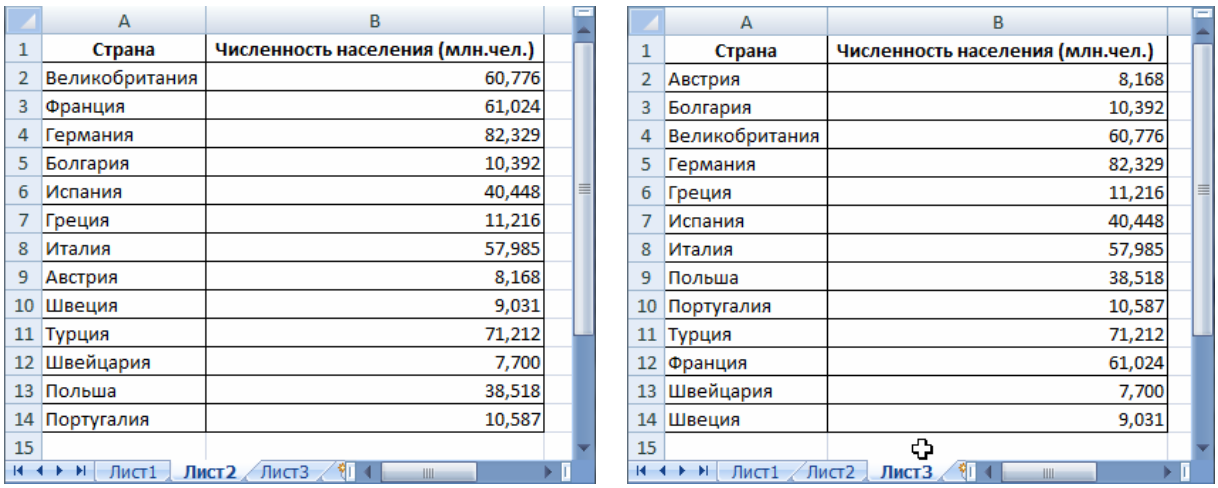


Рис. Л5.4. Результат нажатия кнопок «Добавить запись» и «Отсортировать»

9) написать программный код обработки событий `Click` кнопок «Очистить» и «Выход»;

```
Private Sub clean_Click()
   For i = 1 To 30                    'удаляем содержимое ячеек
     Worksheets("Лист1").Range("A" & i + 14).ClearContents
```

```
    Worksheets("Лист1").Range("B" & i + 1).ClearContents
    Worksheets("Лист2").Range("A" & i).ClearContents
    Worksheets("Лист2").Range("B" & i).ClearContents
    Worksheets("Лист3").Range("A" & i).ClearContents
    Worksheets("Лист3").Range("B" & i).ClearContents
  Next i
End Sub

Private Sub exitForm_Click()
  End
End Sub
```

Сохранение рабочей книги

Сохранить рабочую книгу под именем «VBA_8_Фамилия» (без кавычек) в папке, указанной преподавателем.

ЛАБОРАТОРНАЯ РАБОТА № 6
Двумерные массивы в VBA для Microsoft Excel

Цель выполнения работы: изучение приемов работы с двумерными массивами в Excel VBA.

Теоретические сведения. См. разделы 4.1.9 и 4.1.14 лекций.

Постановка задачи № 1

Известны данные о количестве часов проката каждой из 5 различных марок легковых автомобилей 3 филиалами фирмы.

Необходимо разработать пользовательский интерфейс и составить программу в редакторе Excel VBA, с помощью которой будет выполняться:

1) ввод исходных данных двумерного массива и их вывод в виде таблицы на рабочий лист Microsoft Excel;
2) определение:
   – марки автомобиля, которая по общим часам проката у трех филиалов больше всего уступает другим маркам,
   – среднего количества часов проката одного автомобиля в центральном филиале, количества марок автомобилей центрального филиала, для которых количество часов проката не превышало найденного среднего показателя, и марок таких автомобилей;
3) вывод полученных результатов на тот же рабочий лист.

Решение

1) Открыть новую рабочую книгу MS Excel;
2) на первом рабочем листе создать таблицу и ввести названия пяти марок автомобилей (рис. Л6.1);
3) ниже таблицы разместить три кнопки с помощью соответствующего инструмента панели «Элементы управления» (рис. Л6.1). Свойства размещенных на первом листе кнопок представлены в таблице Л6.1;

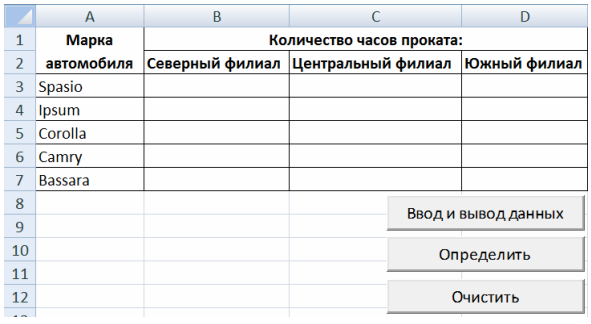


Рис. Л6.1. Количество часов проката легковых автомобилей фирмы

Таблица Л6.1

Объект	Свойство	Объект	Свойство
CommandBut-tonl	Name = inputOutputData Caption = Ввод и вывод данных	CommandBut-ton3	Name = clean Caption = = Очистить
Command-Button2	Name = calc Caption = Определить		

Примечание 5.1. Значения количества часов проката будут вводиться с помощью функции InputBox и присваиваться соответствующим элементам двумерного массива.

4) для обработки событий Click кнопок «Ввод и вывод данных», «Определить» и «Очистить» ввести в модуле «Лист1» приведенный ниже программный код;

```
Dim KolHour(1 To 10, 1 To 10) As Integer
Dim KM, KF, i, j As Integer
Private Sub InputOutputData_Click()
      KM = Val(InputBox("Введите количество марок легковых
   автомобилей", "Ввод данных"))
      KF = Val(InputBox("Введите число филиалов фирмы", "Ввод
   данных"))
      For i = 1 To KM
         For j = 1 To KF
            KolHour(i, j) = Val(InputBox("Введите количество
   часов проката " & Cells(i + 2, 1) & " " & Cells(2, j + 1),
   "Ввод данных"))
            Cells(i+2, j+1) = KolHour(i, j)  'вывод элементов
                                              массива на лист
         Next j
      Next i
End Sub

Private Sub calc_Click()
Dim sH, minS, minMarka, k As Integer
Dim stroka As String
   minS = 32000
   For i = 1 To KM
      sH = 0
      For j = 1 To KF
          sH=sH+KolHour(i, j)   'нахождение суммы часов проката
                                каждой марки
      Next j
```

```
      If sH<minS Then minS=sH: minMarka=I   'поиск минимальной
                                            из этих сумм
      Next i
      MsgBox "Марка автомобиля, уступающая остальным маркам по
   часам проката у трех филиалов, - " & Cells(minMarka + 2, 1)
   'нахождение суммы часов проката по центральному филиалу
   sH = 0
   For i = 1 To KM
      sH = sH + KolHour(i, 2)
   Next i
   MsgBox "Среднее количество часов проката одного автомобиля в
   центральном филиале составляет " & Str(Round(sH / KM, 2))
   'поиск количества марок авто и их наименований с часами
   'проката, не выше среднего по центральному филиалу
   k = 0
   stroka = ""
   For i = 1 To KM
      If KolHour(i, 2) <= sH / KM Then
         k = k + 1
         stroka = stroka + Cells(i + 2, 1) + " "
      End If
   Next i
   MsgBox "Количество марок автомобилей центрального филиала, у
   которых количество часов проката не превышает среднее, " &
   Str(k) & ", а марки: " & stroka
   Cells(KM + 3, 1) = "Марка авто"  'вывод результатов на лист
   Cells(KM + 3, 2) = Cells(minMarka + 2, 1)
   Cells(KM + 4, 1) = "Среднее кол-во часов проката"
   Cells(KM + 4, 3) = sH / KM
   Cells(KM + 5, 1) = "Количество марок"
   Cells(KM + 5, 3) = k
   Cells(KM + 6, 1) = "Марки:"
   Cells(KM + 6, 3) = stroka
End Sub

Private Sub clean_Click()
   For i = 1 To KM + 4
      For j = 1 To KF
         Cells(i + 2, j + 1).ClearContents
      Next j
      Cells(i + KM + 2, 1).ClearContents
   Next i
End Sub
```

5) проверить работу программы (рис. Л6.2).

	A	B	C	D	E
1	**Марка**	**Количество часов проката:**			
2	**автомобиля**	**Северный филиал**	**Центральный филиал**	**Южный филиал**	
3	Spasio	75	42	14	
4	Ipsum	45	12	76	
5	Corolla	28	74	62	
6	Camry	57	68	32	
7	Bassara	48	29	68	
8				Ввод и вывод данных	
9					
10				Определить	
11					
12				Очистить	
13					

	A	B	C	D	E
1	**Марка**	**Количество часов проката:**			
2	**автомобиля**	**Северный филиал**	**Центральный филиал**	**Южный филиал**	
3	Spasio	75	42	14	
4	Ipsum	45	12	76	
5	Corolla	28	74	62	
6	Camry	57	68	32	
7	Bassara	48	29	68	
8	Марка авто	Spasio		Ввод и вывод данных	
9	Среднее кол-во часов проката		45		
10	Количество марок		3	Определить	
11	Марки:		Spasio Ipsum Bassara		
12				Очистить	
13					

Рис. Л6.2. Результат нажатия кнопок «Ввод и вывод данных» и «Определить»

Сохранение рабочей книги

Сохранить рабочую книгу под именем «VBA_8.1_Фамилия» (без кавычек) в папке, указанной преподавателем.

Постановка задачи № 2

Составить программу в редакторе Excel VBA, с помощью которой будет выполняться:

– ввод элементов матрицы размерностью 3×4 и их вывод в ячейки рабочего листа (с точностью до 2 знаков после запятой);

– определение наименьшего по модулю элемента каждого столбца матрицы и произведения элементов матрицы, значения которых по модулю не превышают 1,5;

– вывод полученных результатов на тот же рабочий лист.

Элементы матрицы и результаты вычислений будут выводиться на второй рабочий лист книги MS Excel.

Решение

1) Открыть новую рабочую книгу MS Excel;

2) разместить на втором листе две кнопки с помощью соответствующего инструмента панели «Элементы управления» (рис. Л6.3);

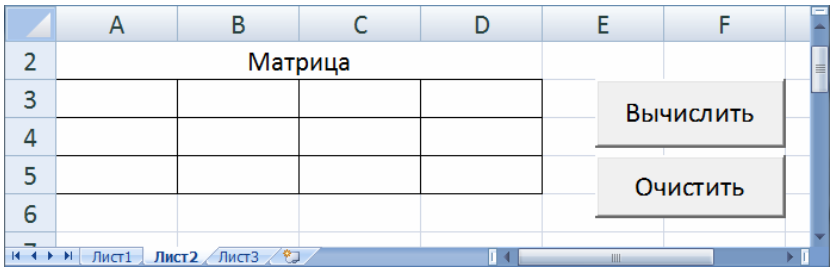


Рис. Л6.3. Матрица

3) изменить свойства этих кнопок в соответствии с данными таблицы Л6.2;

Таблица Л6.2

Объект	Свойство	Объект	Свойство
CommandButton1	Name = calc Caption = Вычислить	CommandButton2	Name = clean Caption = = Очистить

4) для обработки событий Click кнопок «Вычислить» и «Очистить» введем в модуле Лист2 приведенный ниже программный код;

```
Private Sub calc Click()
Dim matr() As Single                'объявление динамического массива
Dim i, j, KolStr, KolStolb, im As Integer
Dim minElem, proizvElem As Single
    KolStr=Val(InputBox("Введите количество строк матрицы",
    "Ввод данных"))
    KolStolb = Val(InputBox("Введите количество столбцов матрицы",
    "Ввод данных"))
    ReDim matr(1 To KolStr, 1 To KolStolb)      'изменение раз-
                                                мера 'массива
    Randomize                        'генератор случайных чисел
    'задание значений элементов матрицы с помощью генератора
    'случайных чисел и их вывод на лист
    For i = 1 To KolStr
      For j = 1 To KolStolb
         matr(i, j) = Round((-1) ^ (i + j) * Rnd * 5, 2)
         Cells(i + 1, j) = matr(i, j)
      Next j
```

```
   Next i
   proizvElem = 1
   For j = 1 To KolStolb
       minElem = Abs(matr(1, j))
       For i = 1 To KolStr
   'поиск минимума по столбцу
      If Abs(matr(i, j))< minElem Then minElem = Abs(matr(i, j)):
      im = i
   'вычисление произведения
      If Abs(matr(i, j))<= 1.5 Then proizvElem = proizvElem *
      matr(i, j)
      Next i
      Cells(6, j) = matr(im, j)
   Next j
   Cells(6, KolStolb + 1) = "|Минимум столбца|"
   Cells(8, KolStolb + 1) = "Произведение |x(i,j)|<=1,5"
   Cells(8, KolStolb) = Round(proizvElem, 4)
End Sub

Private Sub clean_Click()
   For i = 2 To 12
      For j = 1 To 12
         Cells(i, j).ClearContents
      Next j
   Next i
End Sub
```

5) проверить работу программы (рис. Л6.4).

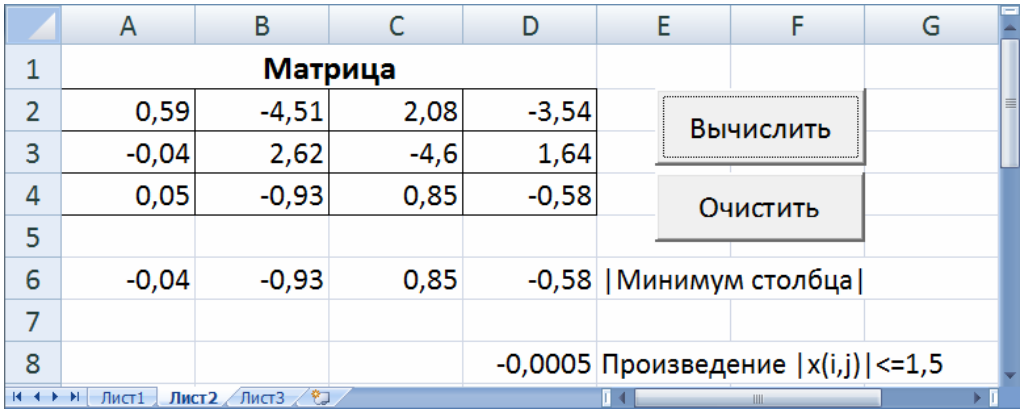


Рис. Л6.4. Результат нажатия кнопки «Вычислить»

Сохранение рабочей книги

Сохранить рабочую книгу под именем «VBA_8.2_Фамилия» (без кавычек) в папке, указанной преподавателем.

ЛАБОРАТОРНАЯ РАБОТА №7
Системы поиска информации в сети Интернет

Цель выполнения работы: приобретение навыков поиска информации в сети Интернет.

Теоретические сведения. См. разделы 2.3.2–2.3.4 лекций.

Просмотр и сохранение Web-страниц

1) Запустить программу Internet Explorer;

2) в поле адресной строки ввести *URL*-адрес *http://www.yandex.ru* – произойдет загрузка титульной страницы поисковой системы Яndex (рис. Л7.1);

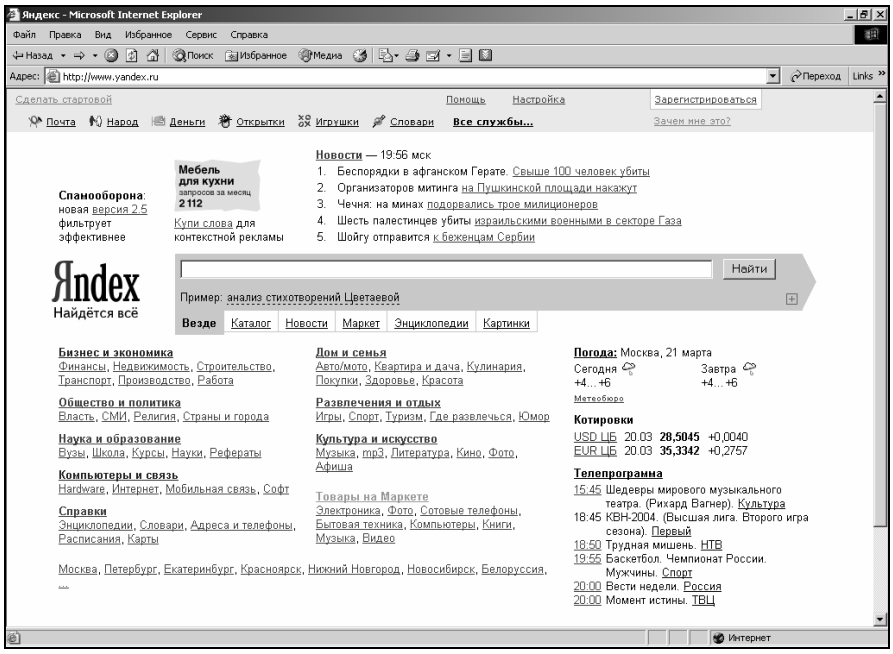


Рис. Л7.1. Титульная страница поисковой системы Яndex

3) найти на титульной странице гиперссылку *Бизнес и экономика* и щелкнуть на этой гиперссылке;

4) в списке ссылок раздела *Бизнес и экономика* найти ссылку *Деловые услуги*. Навести на нее указатель мыши, щелкнуть левой кнопкой и перейти в раздел, посвященный электронной коммерции. Просмотреть содержание раздела (рис. Л7.2);

5) двумя щелчками на кнопке Назад вернуться к титульной странице поисковой системы;

6) двумя щелчками на кнопке  вернуться в раздел электронной коммерции;

7) сохранить текущую Web-страницу на жестком диске. Для этого выполнить команду меню *Файл→Сохранить как* – откроется диалоговое окно *Сохранение Web-страницы*. Здесь можно ввести содержательное имя для сохраняемой страницы, например *Ресурсы по электронной коммерции*. В качестве типа файла выбрать *Веб-страница полностью*. В этом случае страница сохранится вместе со всеми встроенными элементами оформления, (например, рисунками);

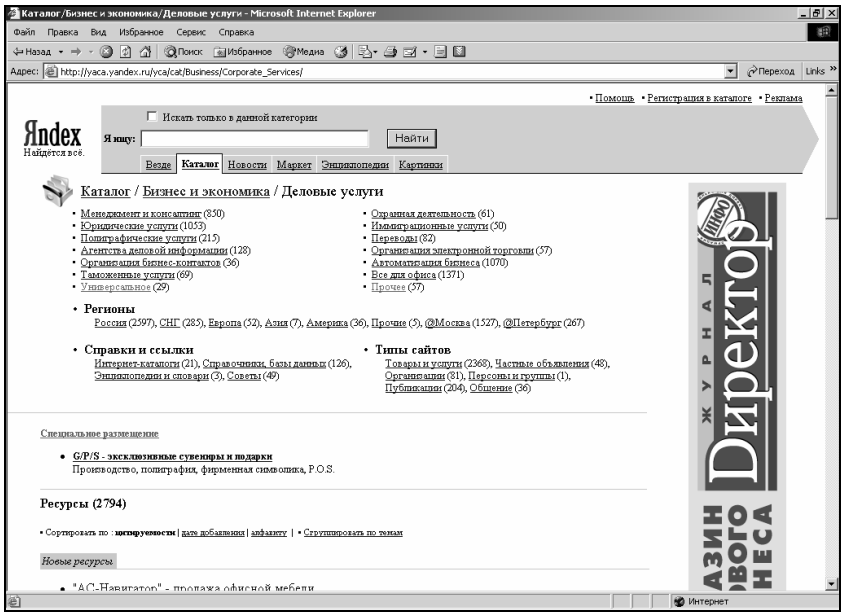


Рис. Л7.2. Содержание раздела «Деловые услуги»

8) в качестве места сохранения Web-страницы назначить папку *C:\Temp*;

9) выполнить команду *Избранное→Добавить в избранное* – откроется диалоговое окно *Добавление в избранное*. Его средства дают нам возможность запомнить *URL*-адрес текущей страницы и в будущем не вводить его в адресной строке браузера. Щелкнуть на кнопке ОК;

10) вывести меню *Избранное* и убедиться в том, что в меню появилась запись со ссылкой на текущую страницу;

11) с помощью кнопки Назад вернуться к предыдущей странице;

12) вывести меню *Избранное* и щелкнуть на только что созданной ссылке. Убедиться, что страница, адрес которой был внесен в список избранных ссылок, загружается немедленно;

13) вывести меню *Избранное* еще раз. Найти только что созданную ссылку. Щелкнуть на ней правой кнопкой мыши и в открывшемся контекстном меню выбрать пункт *Удалить*. Убедиться в том, что ссылка исчезла из списка избранных ссылок.

Поиск информации по ключевым словам

1) Вернуться к титульной странице поисковой системы;

2) на этой странице найти поле для ввода ключевых слов (*Я ищу*) и кнопку запуска поиска Найти. Необходимо будет найти Web-страницы, посвященные электронной коммерции;

3) в поле для ввода ключевых слов ввести *электронная*;

4) щелкнуть на кнопке Найти;

5) просмотреть результаты поиска и обратить внимание на количество найденных Web-страниц (рис. Л7.3);

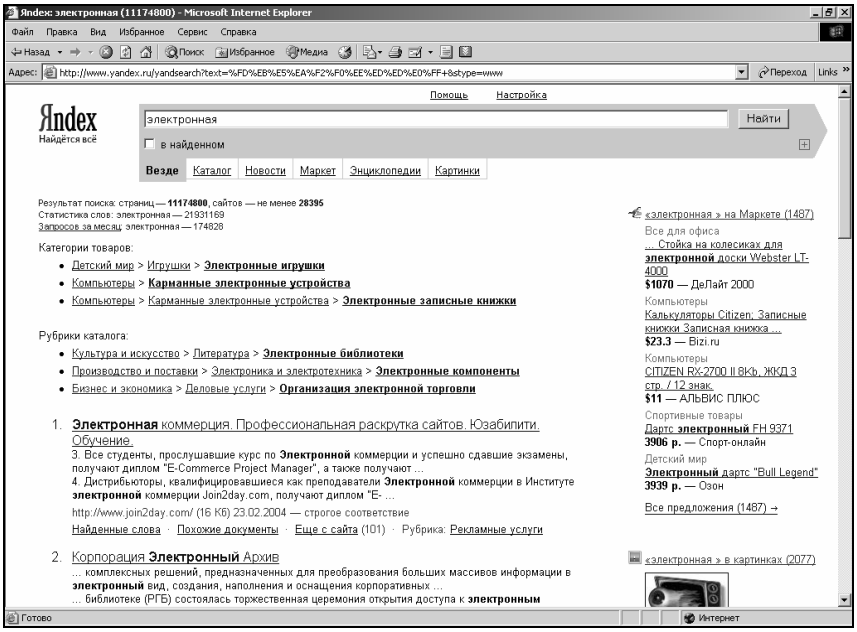


Рис. Л7.3. Результаты поиска по ключевому слову «электронная»

6) в поле для ввода ключевых слов ввести *коммерция* и щелкнуть на кнопке Найти;

7) просмотреть результаты поиска, обратив внимание на количество найденных Web-страниц (см. рис Л7.4);

8) в поле для ввода ключевых слов ввести слова *электронная коммерция* и щелкнуть на кнопке Найти. Обратить внимание на количество найденных Web-страниц (рис. Л7.5). Объяснить, почему количество страниц со словами *электронная коммерция* меньше, чем количество страниц со словом *электронная* и со словом *коммерция*;

9) в поле для ввода ключевых слов ввести слова *электронная OR коммерция* и щелкнуть на кнопке Найти (в данном случае слово *OR* – это логический оператор *ИЛИ*, а не ключевое слово для поиска). Обратить внимание на количество найденных Web-страниц (рис. Л7.6);

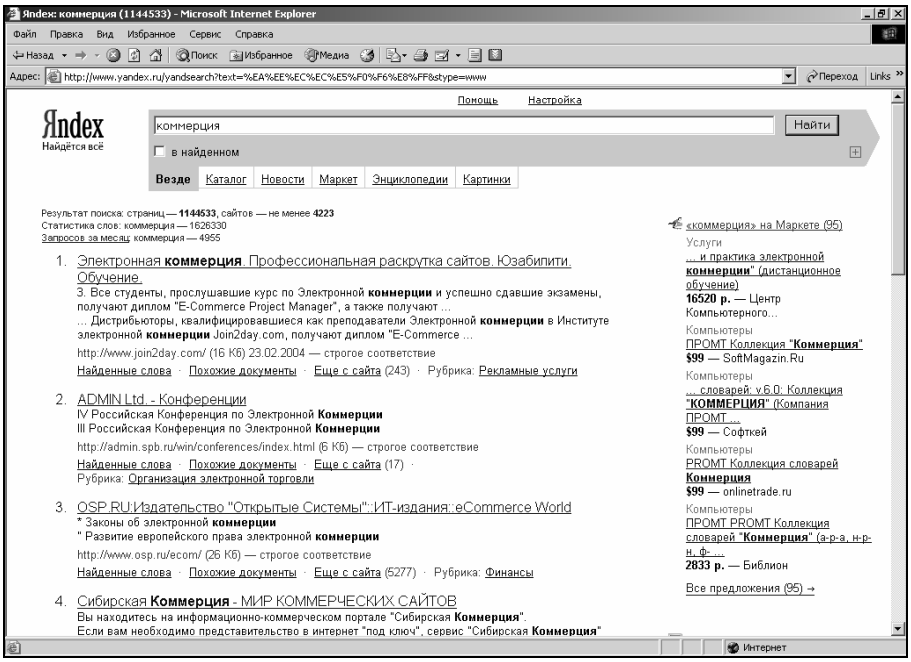


Рис. Л7.4. Результаты поиска по ключевому слову «коммерция»

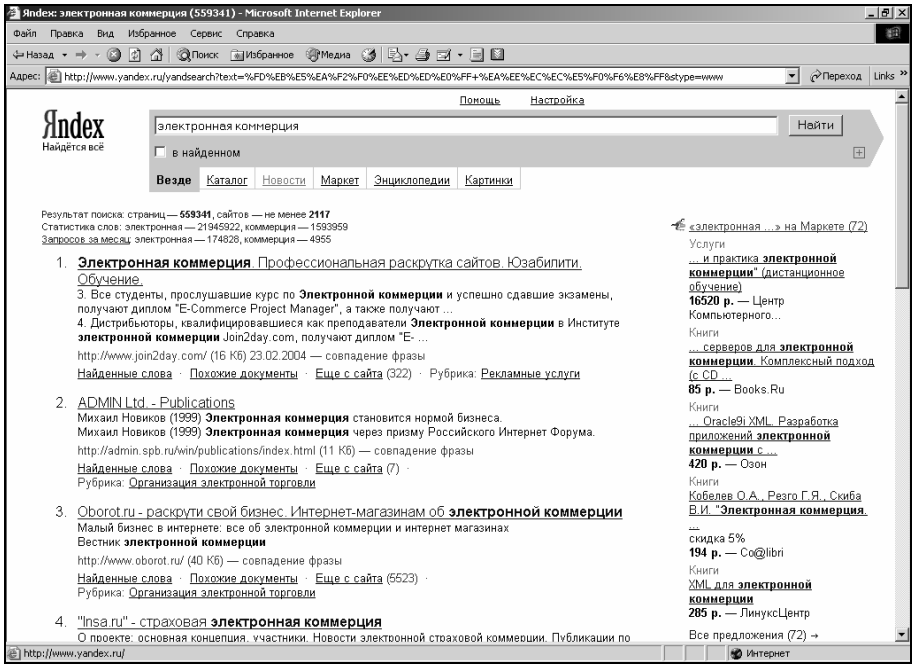


Рис. Л7.5. Результаты поиска по ключевым словам «электронная коммерция»

10) с помощью кнопки Назад вернуться к результату поиска по ключевым словам *электронная коммерция*. Просмотреть список найденных Web-ресурсов. Щелкнуть на гиперссылке, выданной в качестве первой. Дождаться загрузки документа. Оценить его полезность;

11) с помощью кнопки Назад вернуться к предыдущей Web-странице и воспользоваться второй гиперссылкой. Дождаться окончания загрузки документа и оценить его полезность.

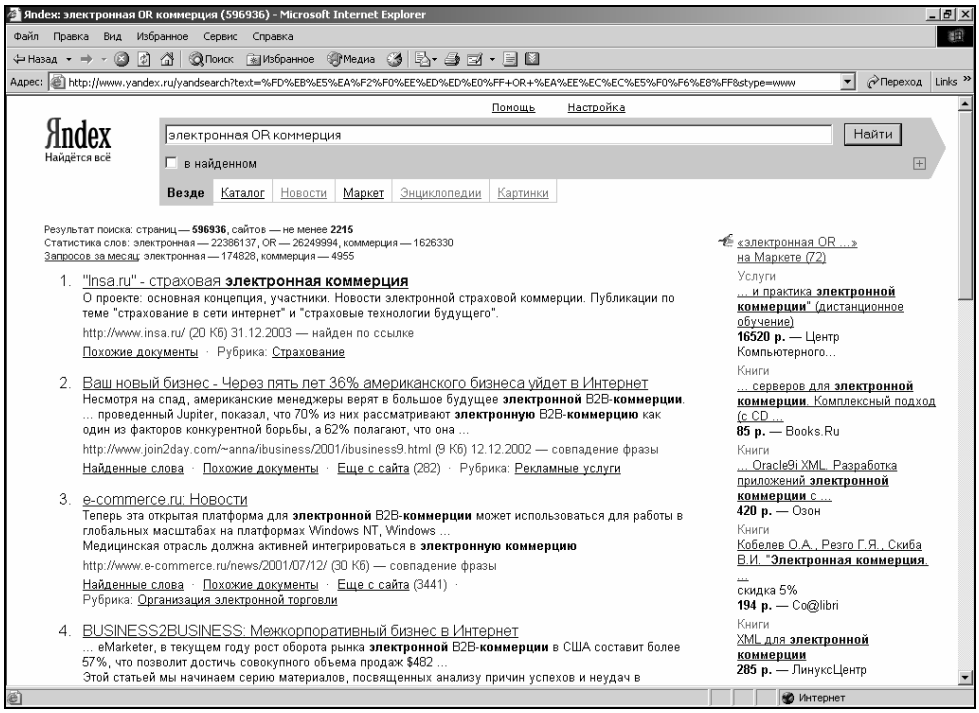


Рис. Л7.6. Результаты поиска по ключевым словам «электронная OR коммерция»

Загрузка файла из Интернета

1) В строке Адрес ввести: ftp://ftp.microsoft.com/;

2) рассмотреть способ представления каталога архива FTP в программе Internet Explorer. Обратить внимание на то, как выглядит значок в строке адреса (рис. Л7.7);

3) двойными щелчками на значках папок открыть папку /Products/Windows/ Windows95/CDRomExtras/FunStuff/ (рис Л7.8).

4) щелкнуть на значке *clouds.exe* правой кнопкой мыши и выбрать в контекстном меню пункт *Копировать в папку*;

5) выбрать папку, специально отведенную для хранения загруженных файлов, и задать имя файла;

6) установить в диалоговом окне загрузки файла флажок *Закрыть диалоговое окно после завершения загрузки*;

7) следить за ходом загрузки файла по этому диалоговому окну (рис Л7.9);

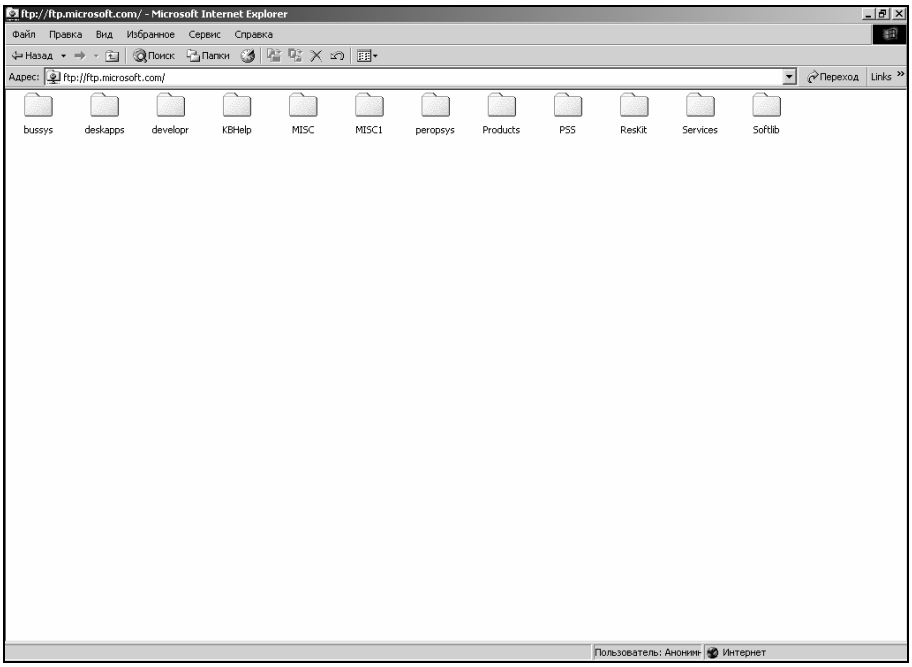


Рис. Л7.7. Представления каталога архива FTP в программе Internet Explorer

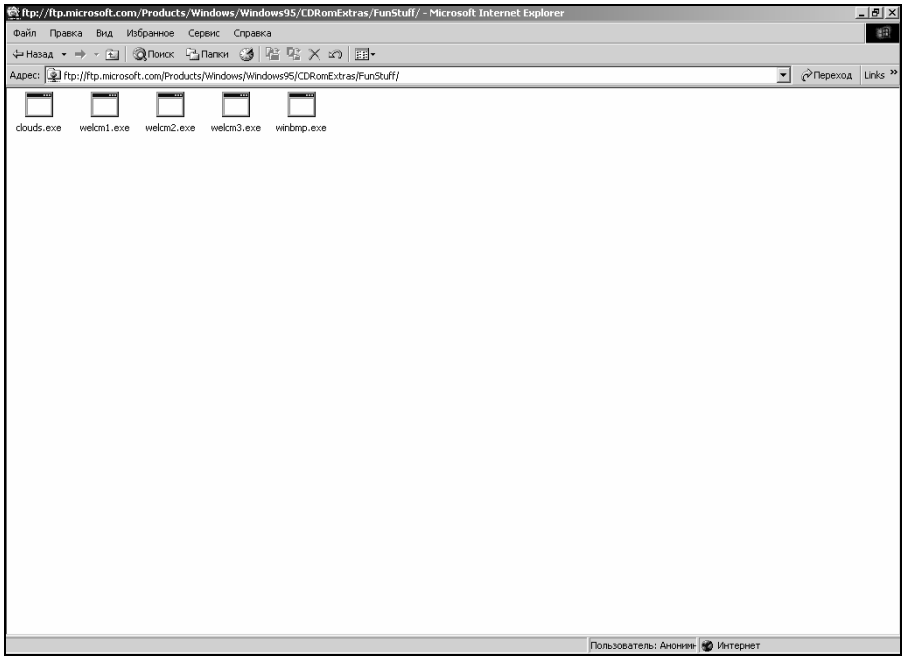


Рис. Л7.8. Содержимое папки FunStuff

8) когда загрузка файла завершится, закрыть диалоговое окно, информирующее о завершении загрузки;

9) открыть папку, в которой был сохранен загруженный файл, при помощи программы *Проводник*;

10) убедиться, что загруженный файл можно использовать в соответствии с его назначением.

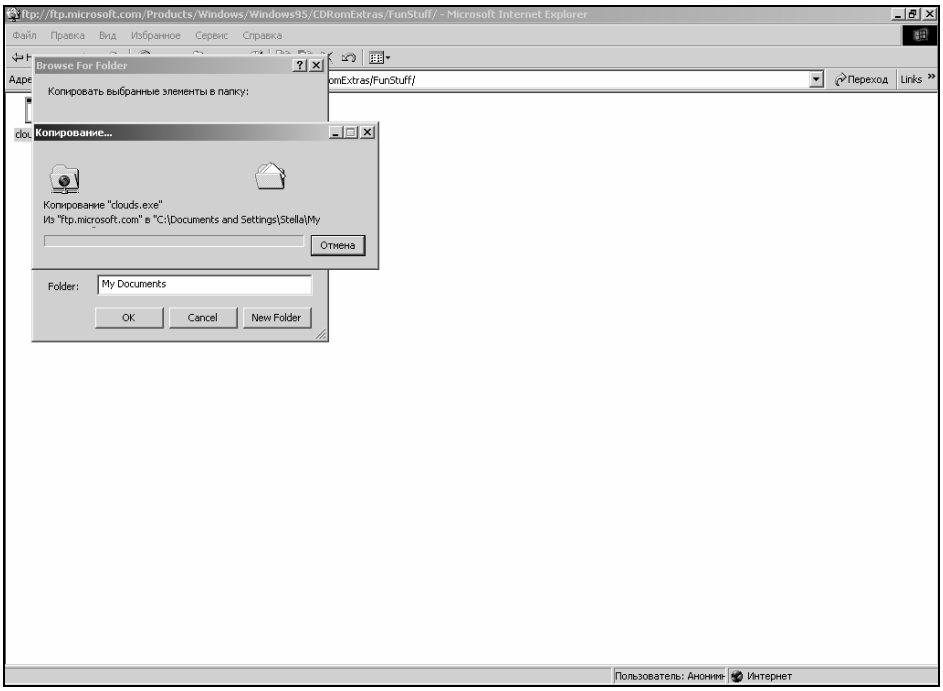


Рис. Л7.9. Процесс загрузки файла с узла FTP

Настройка отображения объектов

1) В строке *Адрес* ввести: *http://www.emoney.ru/eng/menu.asp*. Произойдет подключение к российскому Web-узлу, посвященному электронной коммерции и платежным системам, используемым в Интернете (рис. Л7.10);

2) обратить внимание на время загрузки страницы;

3) посмотреть, как выглядит загруженная страница;

4) выполнить команду меню *Сервис→Свойства обозревателя* – откроется окно диалога *Свойства обозревателя*;

5) открыть вкладку *Дополнительно*;

6) сбросить флажки *Воспроизводить анимацию*, *Воспроизводить звуки*, *Воспроизводить видео* и *Отображать рисунки*;

7) выбрать вкладку *Общие*;

8) щелкнуть на кнопке *Удалить файлы*, подтвердить удаление;

9) щелкнуть на кнопке ОК;

10) щелкнуть на кнопке Обновить;

11) обратить внимание на уменьшение времени загрузки страницы;

12) сравнить внешний вид страницы при предыдущей и нынешней загрузке (рис. Л7.11);

13) щелкнуть на одной из пустых рамок для рисунков правой кнопкой мыши и выбрать в контекстном меню команду *Показать рисунок*.

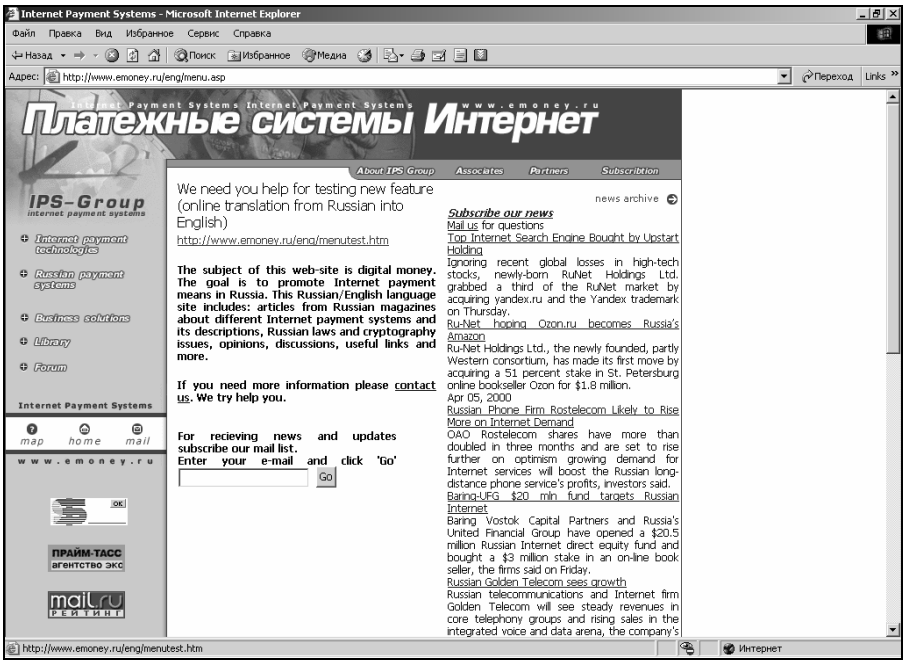


Рис. Л7.10. Титульная страница Web-узла компании IPS-Group

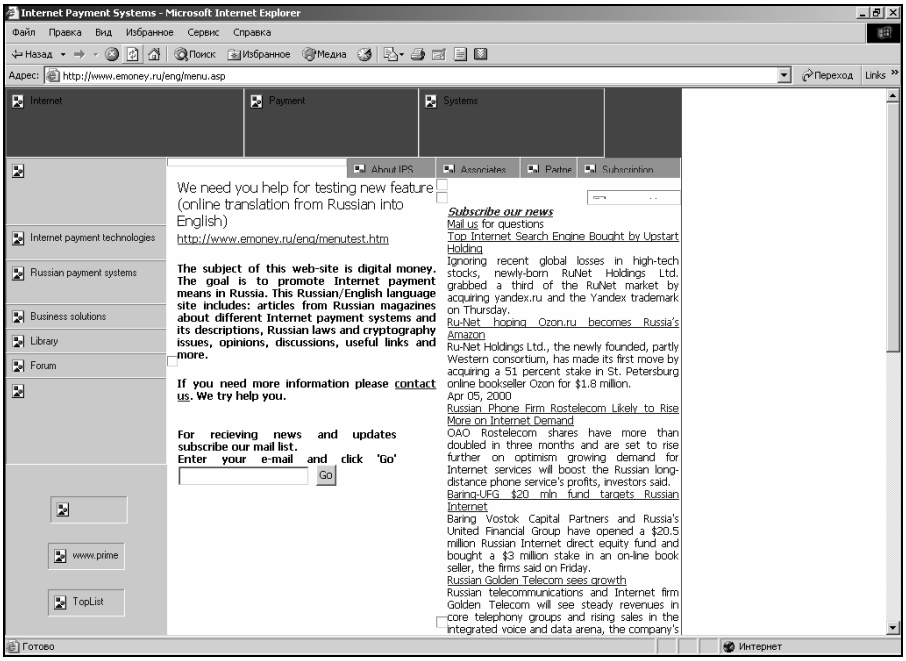


Рис. 7.11. Титульная страница Web-узла компании IPS-Group после отключения отображения рисунков

ЛАБОРАТОРНАЯ РАБОТА № 8
Создание простого HTML-документа

Цель выполнения работы: изучение общей структуры HTML-документа и основных приемов форматирования текста в HTML-документах.

Теоретические сведения

Базовые понятия HTML

HTML-документ – это обычный текстовый документ, созданный в любом текстовом редакторе, например в *Блокноте*, и оформленный в соответствии с правилами языка HTML. Для файлов, содержащих HTML-документы, используется расширение *.html* или *.htm*.

При открытии HTML-документа браузер распознает тэги и учитывает их при отображении текста. Если по каким-то причинам (например, при ошибке в записи тэга) тэг не распознается браузером, то он игнорируется.

Элементы

Элемент – это конструкция языка HTML, или *контейнер*, содержащий данные. HTML включает в себя разные типы элементов, которые позволяют задавать абзацы, гипертекстовые ссылки, таблицы, списки, изображения и т.д. Конструкция `<I> Привет! </I>` представляет собой элемент.

Обычно элемент можно разделить на три части:

– первая часть, `<I>`, называется открывающим тэгом;

– содержание элемента, которое в данном случае состоит из слова `Привет!`;

– третья часть, `</I>` является закрывающим тэгом.

Название элемента (`I`) присутствует и в открывающем и в закрывающем тэге. Оно нечувствительно к регистру, однако в соответствии с соглашениями, принятыми большинством разработчиков, записано в верхнем регистре. Открывающий и закрывающий тэги нужны для указания начала и конца элемента.

Тэги всегда начинаются `<` и заканчиваются символом `>`. В закрывающем тэге перед его именем помещается символ `/`.

Для некоторых типов элементов допускается отсутствие закрывающего тэга. Бывают элементы, не имеющие закрывающего тэга, т.е. его не просто можно опустить, а он вообще не существует в языке.

Атрибуты

Элементы могут содержать параметры, называемые *атрибутами*, которые могут иметь определенные значения (по умолчанию или устанавливаемые авторами). Каждому атрибуту может быть присвоено значение определенного типа.

В начальном тэге элемента может быть указано любое количество допустимых пар атрибут/значение, разделенных пробелами, например:

```
<BODY bgcolor=″#FF000″ text =″#0000FF″>
```

В приведенном примере устанавливается красный цвет фона страницы и синий цвет основного текста. При установке значений нескольких атрибутов порядок их записи не имеет значения.

```
<BODY text =″#0000FF″ bgcolor=″#FF000″ >
```

Регистр тоже не имеет значения при написании названия атрибута, но для повышения читабельности HTML-кода названия атрибутов обычно записываются в нижнем регистре.

Вложенные элементы

Элемент, находящийся внутри другого элемента, называется вложенным. При использовании вложенности следует помнить, что вложенные элементы должны закрываться до того, как будут закрыты внешние элементы.

Структура HTML- документа

В идеальном случае HTML-документ состоит из:

– заголовка документа;
– тела документа.

HTML-документ

```
<HTML>
<HEAD>
<TITLE> HTML- документ </TITLE>
</HEAD>
<BODY>
<H1> Заголовок </H1>
<P> Первый абзац
<P> Второй абзац
</BODY>
</HTML>
```

Заголовок

В заголовке (его еще называют *«шапкой»*) HTML-документа содержатся сведения о документе:

– название (тема документа);
– ключевые слова (используются поисковыми машинами);
– ряд других данных, которые не являются содержимым документа.

Элемент HEAD

Заголовок HTML-документа содержится в элементе HEAD. Для задания этого элемента используются парные тэги <HEAD> и </HEAD>. Все, что находится между двумя указанными тэгами, относится к заголовку HTML-документа. Тэги <HEAD> и </HEAD> не являются обязательными, но лучше их указывать, чтобы можно было легко определить, что именно относится к заголовку HTML-документа.

Элемент TITLE

В самом простом случае в заголовке документа содержится только один элемент – TITLE. Элемент TITLE используется для задания названия HTML-документа. Он задается при помощи парных тэгов <TITLE> и </TITLE>, причем только один раз. Текст, находящийся между этими тэгами, воспринимается и отображается браузерами (и поисковыми машинами) как название документа. Название текста должно быть кратким, но информативным и должно адекватно отображать содержание документа.

Тело HTML- документа

Вся содержательная часть HTML-документа находится в его теле (элемент BODY). Для определения этого элемента используются парные тэги <BODY> и </BODY>. Эти тэги не являются обязательными, но их наличие значительно улучшит наглядность структурной организации HTML-документа, позволит четко отделить его содержимое от заголовка. Все, что помещено между тэгами <BODY> и </BODY>, является содержимым документа, показываемым браузером пользователю.

Наиболее часто используемые атрибуты элемента BODY:

– background – URI, указывающий расположение изображения для фона (обычно берется небольшое изображение, которое размножается для заполнения фона всего документа);

– bgcolor – цвет фона HTML-документа;
– text – цвет шрифта документа;
– link – цвет непосещенных гиперссылок;
– vlink – цвет посещенных гиперссылок;
– alink – цвет гиперссылок при выборе их пользователем.

Задание цветов в элементе BODY

Таблица Л8.1.

Идентификаторы и значения часто используемых цветов

Название	Значение	Цвет
black	#000000	Черный
silver	#808080	Темно-синий
gray	#C0C0C0	Серый
white	#FFFFFF	Белый
maroon	#800000	Бордовый
red	#FF0000	Красный
purple	#800080	Фиолетовый
fuchsia	#FF00FF	Лиловый
green	#00F000	Зеленый
lime	#00FF00	Ярко-зеленый
olive	#808000	Оливковый
yellow	#FFFF00	Желтый
nary	#000080	Темно-синий
blue	#0000FF	Синий
teal	#008080	Сине-зеленый
aqua	#00FFFF	Бирюзовый

Задание начертания текста

Задание начертания текста является простым средством форматирования содержимого документа, которое доступно в HTML. Для изменения начертания текста в HTML-код вводятся элементы, обозначающие текст, написанный с соответствующим начертанием. Для принудительного перехода на следующую строку можно использовать одиночный тэг
.

Таблица Л8.2

Элементы, определяющие начертания текста

Элемент	Описание
B	Полужирное начертание текста
I	Курсивное начертание текста
U	Подчеркнутый текст
STRIKE, S	Перечеркнутый текст
BIG	Текст с увеличенным размером шрифта
SMALL	Текст с уменьшенным размером шрифта
SUP	Верхний индекс
SUB	Нижний индекс
TT	Текст, записанный моноширинным шрифтом (все символы имеют одинаковую ширину)
BLINK	Мерцающий текст (редко поддерживается браузером)

Шрифтовое оформление текста

Если нужно отобразить текст с использованием определенного шрифта, а не применяемого браузером по умолчанию, то в HTML предусмотрен элемент FONT. Он вводится при помощи парных тэгов <FONT> и </FONT>.

Параметры шрифта для элемента FONT устанавливаются с помощью следующих атрибутов:

– face – задает название шрифта (возможные значения можно посмотреть в списке шрифтов в любом текстовом редакторе);

– size – задает размер шрифта (значение от 1 до 7, по умолчанию используется значение 3);

– color – задает цвет шрифта.

Заголовки

В HTML поддерживаются шесть видов заголовков. Им соответствуют элементы H1, H2, H3, H4, H5, H6. Номера определяют уровни заголовков от наиболее важного (1) до наименее важного (6).

Элементы H1 – H6 задаются при помощи парных тэгов. Например, для задания заголовка пятого уровня можно применить следующий код:

<H5> Текст заголовка пятого уровня </H5>

Для заголовков можно задать свойства:

– `align` – выравнивание текста заголовка (по умолчанию используется левостороннее выравнивание);

– `title` – текст подсказки.

Дополнительные теоретические сведения содержатся в разделах 4.1.9 и 4.1.14 лекций.

Задание №1. Знакомство со структурой HTML-документа

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ8_1.HTML или ФАМИЛИЯ8_1.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л8.1).

```
<HTML>
    <HEAD>
      <TITLE>Простой HTML- документ </TITLE>
    </HEAD>
    <BODY>
      <H1> Заголовок </H1>
      Текст страницы
    </BODY>
</HTML>
```

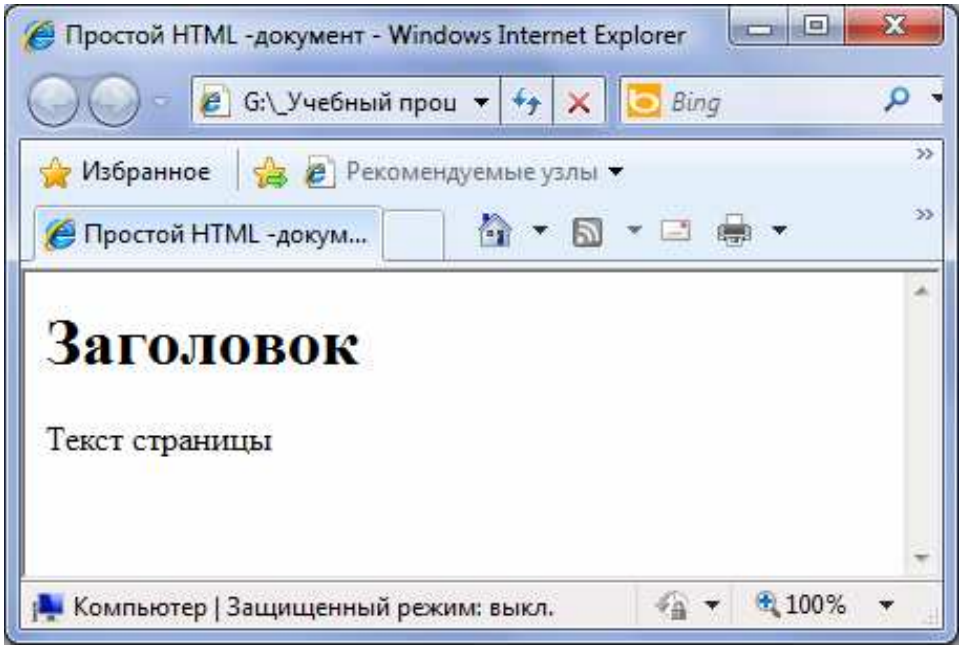


Рис. Л8.1. Результат выполнения задания № 1

Задание №2. Цвета в HTML

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ8_2.HTML или ФАМИЛИЯ8_2.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л8.2).

```
<HTML>
      <HEAD>
         <TITLE> Задание цветов в элементе BODY </TITLE>
      </HEAD>
      <BODY
         Text=”black” (черный)
         Link=”0080FF” (голубой)
         Vlink=”red” (красный)
         Alink=”navy”(темно-синий)>
         Обычный неформатированный текст должен
         отображаться черным цветом
         <P> <A HREF=”ref1”> Непосещенная гиперссылка
      (голубой цвет)</A>
         <P> <A HREF=”ref2”> Посещенная гиперссылка
      (красный цвет)</A>
         <P> <A HREF=”ref3”> Выделенная гиперссылка
      (темно-синий цвет)</A>
</BODY>
</HTML>
```

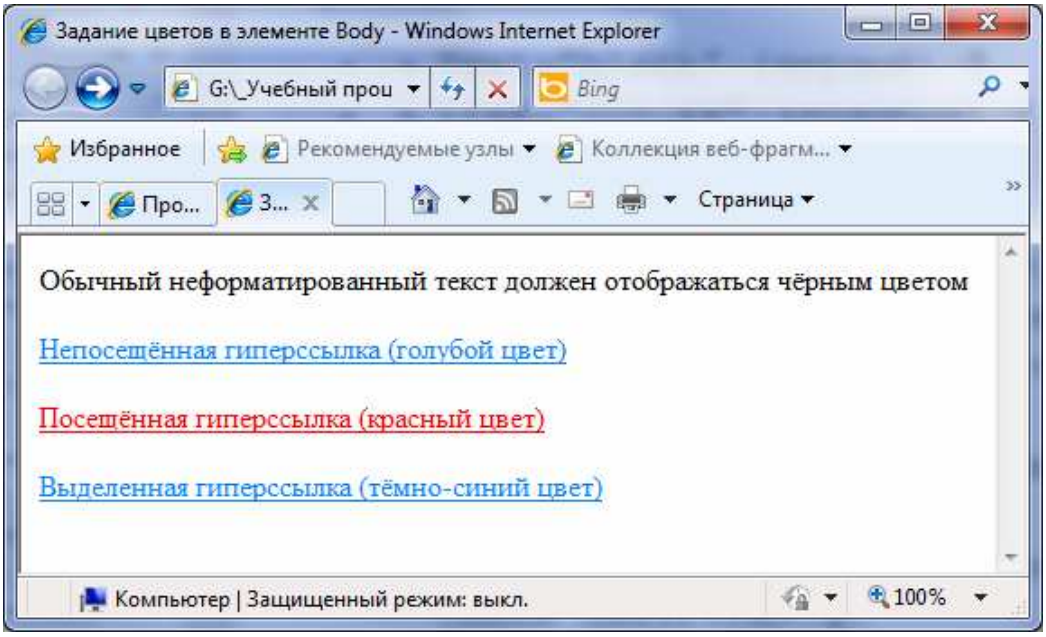


Рис. Л.8.2 Результат выполнения задания № 2

Задание №3. Форматирование текста

Набрать в *Блокноте* приведенный ниже пример. Сохранить с именем ФАМИЛИЯ8_3.HTML или ФАМИЛИЯ8_3.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л8.3).

```
<HTML>
      <HEAD>
         <TITLE> Задание начертания текста </TITLE>
      </HEAD>
      <BODY
         <B> Полужирный текст</B> <BR>
         <I> Курсив</I><BR>
         <U> Подчеркнутый текст</U> <BR>
         <S> Зачеркнутый текст</S> <BR>
         <BIG> Текст увеличенного размера</BIG> <BR>
         <SMALL>Текст уменьшенного размера</SMALL> <BR>
         <SUP> Верхний индекс</SUP> Текст<SUB> Нижний
         индекс</SUB> <BR>
         <TT> Текст, записанный моноширинным шрифтом</TT>
         <BR>
      </BODY>
</HTML>
```

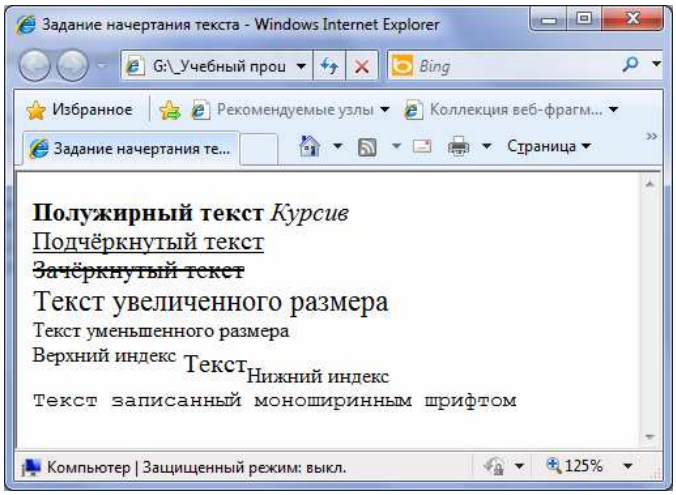


Рис. Л8.3. Результат выполнения задания № 3

Задание №4. Размеры шрифтов в HTML

Набрать в *Блокноте* приведенный ниже пример. Сохранить с именем ФАМИЛИЯ8_4.HTML или ФАМИЛИЯ8_4.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л8.4).

```
<HTML>
     <HEAD>
        <TITLE> Задание шрифта текста </TITLE>
     </HEAD>
     <BODY>
        <FONT face= “Arial”>
        <FONT size=1>
        Самый маленький текст шрифта Arial <BR>
        </FONT>
        <FONT size=7>
        Самый большой текст шрифта Arial<BR>
        </FONT>
        <FONT size=3>
        Обычный текст шрифта Arial <BR>
        </FONT>
        </FONT>
        <FONT face= “Times New Roman”><BR>
        <FONT size=1>
        Самый маленький текст шрифта Times New Roman
        <BR>
        </FONT>
        <FONT size=7>
        Самый большой текст шрифта Times New Roman <BR>
        </FONT>
        <FONT size=3>
        Обычный текст шрифта Times New Roman <BR>
        </FONT>
        </FONT>
     </BODY>
</HTML>
```

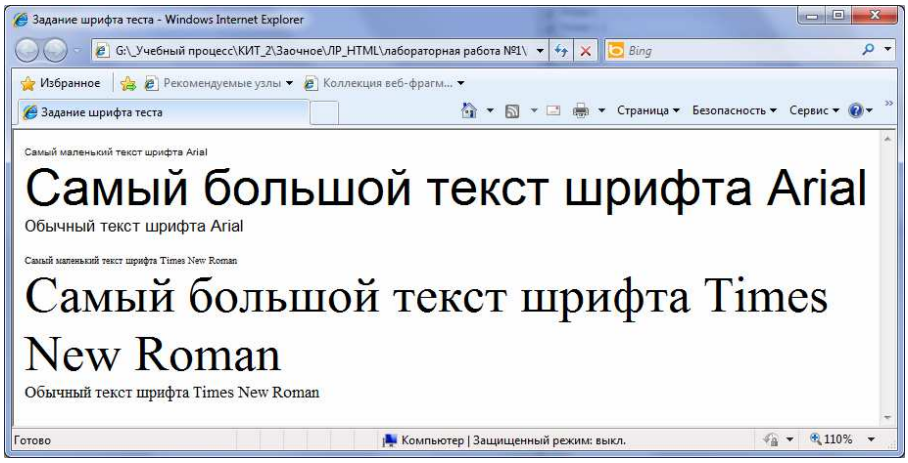


Рис. Л8.4. Результат выполнения задания № 4

Задание №5. Выравнивание абзацев

Набрать в *Блокноте* приведенный ниже пример. Сохранить с именем ФАМИЛИЯ8_5.HTML или ФАМИЛИЯ8_5.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л8.5).

```
<HTML>
     <HEAD>
        <TITLE> Выравнивание абзацев </TITLE>
     </HEAD>
     <BODY>
        <P align=“right”>
        Абзац выровнен вправо </P>
        <P align=“center”>
        Абзац выровнен по центру </P>
        <P align=“justify”>
        Абзац выровнен по ширине </P>
        <P align=“left”>
        Абзац выровнен влево </P>
        <P><NOBR> Этот текст всегда должен оставаться в
        одной строке </NOBR>
     </BODY>
</HTML>
```

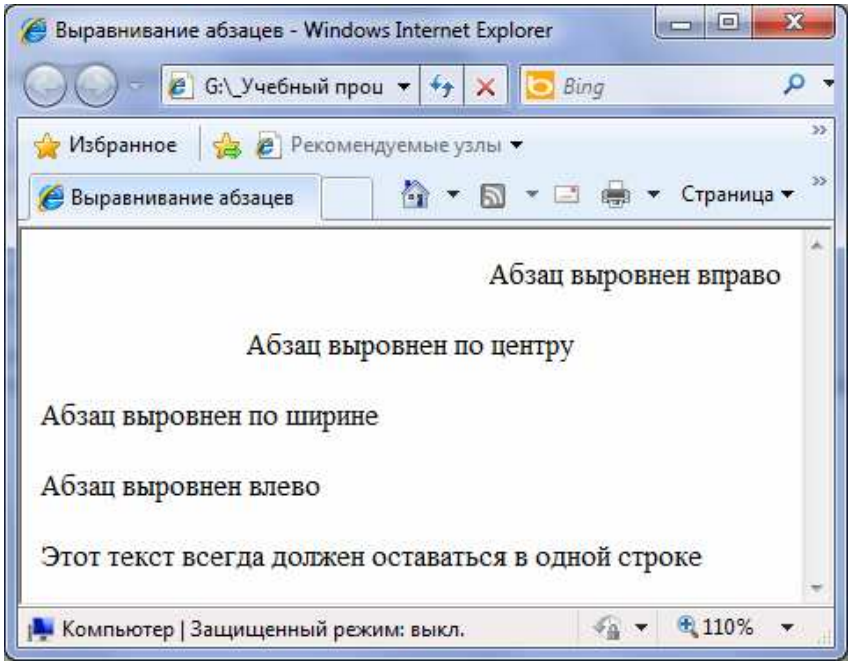


Рис. Л8.5. Результат выполнения задания № 5

Задание 6. Разбиение текста на абзацы

Набрать в *Блокноте* приведенный ниже пример. Сохранить с именем ФАМИЛИЯ8_6.HTML или ФАМИЛИЯ8_6.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л8.6).

```
<HTML>
      <HEAD>
         <TITLE> Разбиение текста на абзацы</TITLE>
      </HEAD>
      <BODY>
         <P title="Первый абзац">
         Неформатированный текст
         <P align= "right" title="Второй абзац">
         Текст с <B> изменением <I> начертания </I> </B>
         <P align= "center" title="Третий абзац">
         <FONT size="+2" face="arial"> Текст с измененным
         шрифтом</FONT>
         <P align= "justify" title="Четвертый абзац">
         Текст этого абзаца автоматически выравнивается
      по ширине справа и слева при переносе слов.
      </BODY>
</HTML>
```

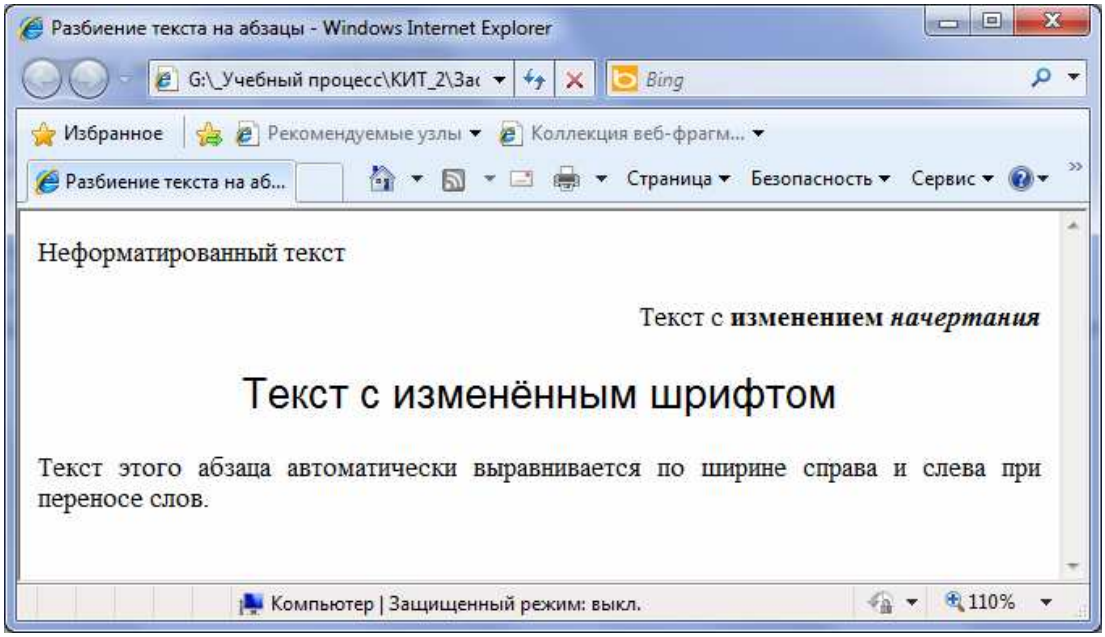


Рис. Л8.6. Результат выполнения задания № 6

Задание 7. Заголовки в HTML

Набрать в *Блокноте* приведенный ниже пример. Сохранить с именем ФАМИЛИЯ8_7.HTML или ФАМИЛИЯ8_7.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л8.7).

```
<HTML>
      <HEAD>
         <TITLE> Заголовки разных уровней</TITLE>
      </HEAD>
      <BODY>
         <H1> Так выглядит заголовок 1-го уровня</H1>
         <H2> Так выглядит заголовок 2-го уровня</H2>
         <H3> Так выглядит заголовок 3-го уровня</H3>
         <H4> Так выглядит заголовок 4-го уровня</H4>
         <H5> Так выглядит заголовок 5-го уровня</H5>
         <H6> Так выглядит заголовок 6-го уровня</H6>
      </BODY>
</HTML>
```

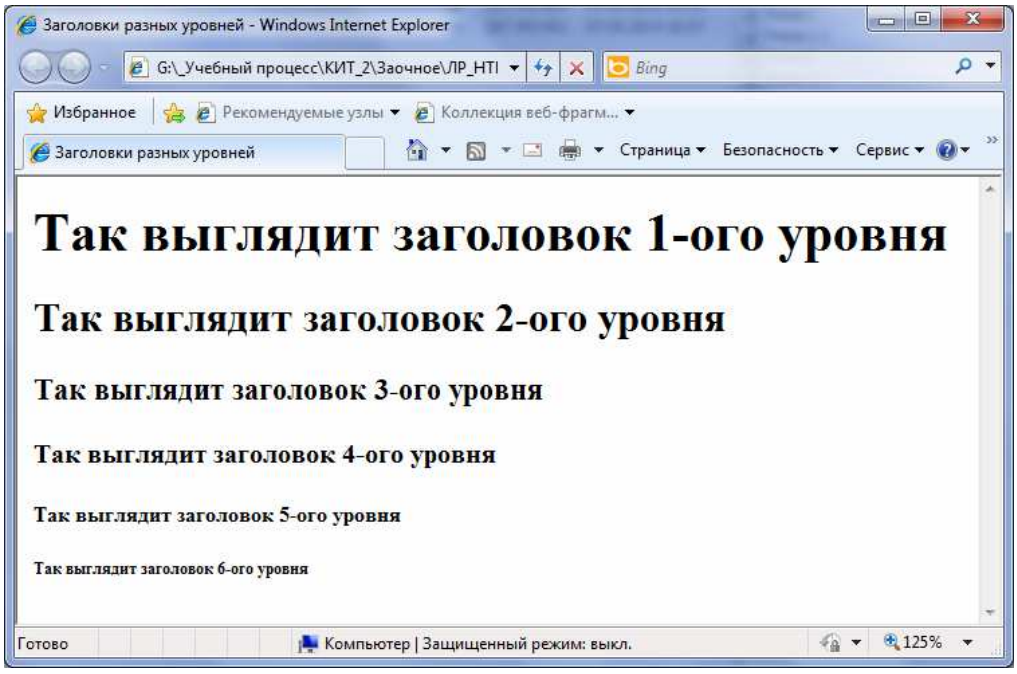


Рис. Л8.7. Результат выполнения задания № 6

ЛАБОРАТОРНАЯ РАБОТА № 9
Списки в HTML-документах

Цель выполнения работы: изучение основных приемов создания списков в HTML-документах.

Теоретические сведения

В HTML поддерживаются три вида списков:
- маркированные,
- нумерованные,
- списки определений.

Маркированные списки

Маркированные списки применяются для перечисления неупорядоченной информации. В таком списке каждый новый элемент выделяется маркером (отсюда и название списка). В HTML для обозначения маркированного списка используется элемент UL, который задается парными тегами <UL> и </UL> между тегами помещаются элементы списка.

Текст элементов списка начинается после тега <LI> и заканчивается тегом </LI>. С закрывающим тегом </LI> ситуация такая же как и с закрывающим тегом </P>: его использование нежелательно. Если тег </LI> опустить, то текстом элемента списка считается весь текст, расположенный до следующего тега <LI> или до закрывающегося тега </UL>.

Пример простого маркированного списка, состоящего из трех элементов (рис. Л9.1).

```
<UL>
<LI> Первый элемент
<LI> Второй элемент
<LI> Третий элемент
</UL>
```

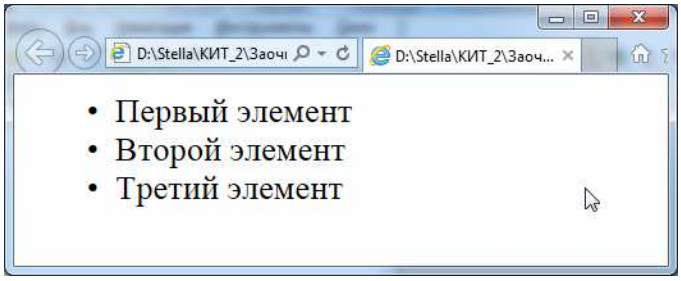


Рис. Л9.1. Пример простого маркированного списка, состоящего из трех элементов

К тексту элементов любых списков можно применять рассмотренные ранее средства HTML по форматированию текста. Элементы UL и LI имеют ряд атрибутов. Специфичными для элемента UL являются следующие атрибуты:

1) compact – заставляет браузер показывать список более компактно (действие этого атрибута зависит от конкретного браузера);

2) type – позволяет задать тип маркера списка, может принимать значения:

– circle (круг без заливки),

– dict (круг с заливкой),

– square (квадрат).

Атрибут type можно указывать также для нужных элементов списка LI, если понадобится изменить тип маркера только некоторых элементов списка.

Нумерованные списки

Нумерованные списки применяются для упорядочения приводимых данных. При нумерации элементов таких списков могут быть использованы как арабские, так и римские цифры, символы латинского алфавита.

Нумерованный список в тексте HTML-документа обозначается элементом OL при помощи парных тегов <OL> и </OL>. Элементы нумерованного списка задаются в точности так же, как и элементы маркированного списка.

Нумерованный список несколько отличается от маркированного не только внешним видом, но и набором атрибутов и их вложенными значениями:

3) compact – заставляет браузер отображать список компактно;

4) type – задает тип нумерации элементов списка, доступные значения:

– 1 (используются арабские цифры, по умолчанию),

– I или i (большие или малые римские цифры),

– A или a (большие или малые буквы латинского алфавита);

– start – номер первого элемента списка (при задании start нужно учитывать тип нумерации элементов списка, например номеру 5 соответствует латинская буква E). Атрибут start часто используется, когда нужно продолжить нумерацию предшествующего списка после отрывка текста, не являющегося элементом ни одного списка (например, после пояснения элемента предыдущего списка).

Список определений

Список определений задается внутри элемента DL (для его задания используются парные теги <DL> </DL>). Каждый элемент списка определений состоит из двух частей: из термина (HTML-элемент DT) и определения термина (HTML-элемент DD).

Линейки

Линейка – горизонтальная линия в окне браузера. Для вставки линейки в HTML-документ используется элемент HR. Этот HTML-элемент задается при помощи одиночного тега <HR> и имеет следующие атрибуты:

– align – задает выравнивание линейки в окне браузера, может принимать значения left, right, или center.

– noshade – булев атрибут, указывает браузеру, что линейку следует отображать плоской (без традиционной тени);

– size – численное значение, определяющее толщину линии;

– width – численное значение, определяющее ширину линейки.

Значения size и width могут задаваться как абсолютные (в пикселях) или как относительные (в процентах от ширины окна браузера) величины.

По умолчанию используется выравнивание по центру и ширина линейки, равная 100% от ширины окна браузера.

Задание № 1. Создание маркированных списков в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ9_1.HTML или ФАМИЛИЯ9_1.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л9.2).

```
<HTML>
   <HEAD>
       <TITLE> Маркированные списки </TITLE>
   </HEAD>
   <BODY>
       Список с закрашенными круглыми маркерами
          <UL>
          <LI> Первый элемент
          <LI> Второй элемент
          </UL>
       Список с незакрашенными круглыми маркерами
```

```
            <UL type =“circle”>
            <LI> Первый элемент
            <LI> Второй элемент
            </UL>
         Список с квадратными маркерами
            <UL type =“square”>
            <LI> Первый элемент
            <LI> Второй элемент
            </UL>
         Список с разными маркерами элементов
            <UL>
            <LI>Закрашенный круг
            <LI type =“circle”> Окружность (type=circle)
            <LI type =“square”> Квадрат (type=square)
            </UL>
   </BODY>
</HTML>
```

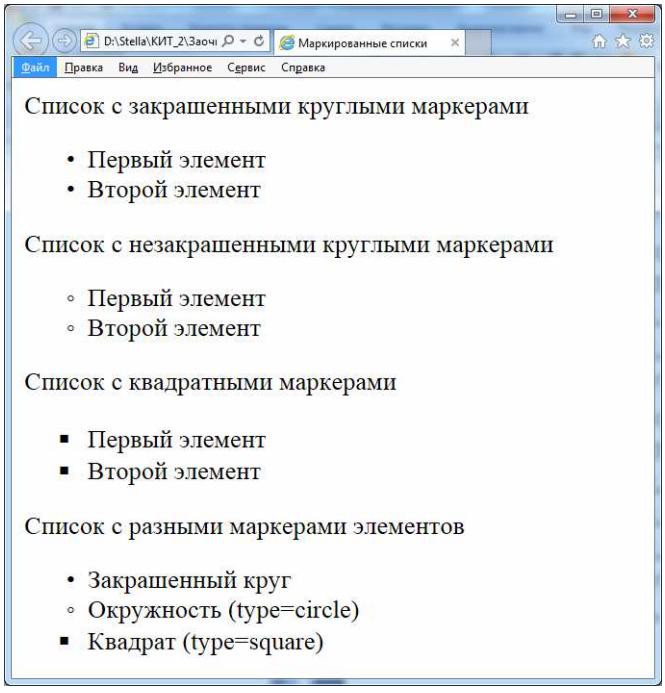


Рис. Л9.2. Результат выполнения задания № 1

Задание № 2. Создание нумерованных списков в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ9_2.HTML или ФАМИЛИЯ9_2.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л9.3).

```
<HTML>
   <HEAD>
      <TITLE> Нумерованные списки </TITLE>
   </HEAD>
   <BODY>
      Нумерация арабскими цифрами
         <OL>
         <Li>Первый элемент
         <Li>Второй элемент
         </OL>
      Продолжение нумерации, но большими римскими
      цифрами
         <OL type=“I” start=3>
         <Li>Третий элемент
         <Li>Четвертый элемент
         </OL>
      Новый список, нумерация большими латинскими
      буквами
         <OL type=“A”>
         <Li> Первый элемент
         <Li> Второй элемент
         </OL>
   </BODY>
</HTML>
```

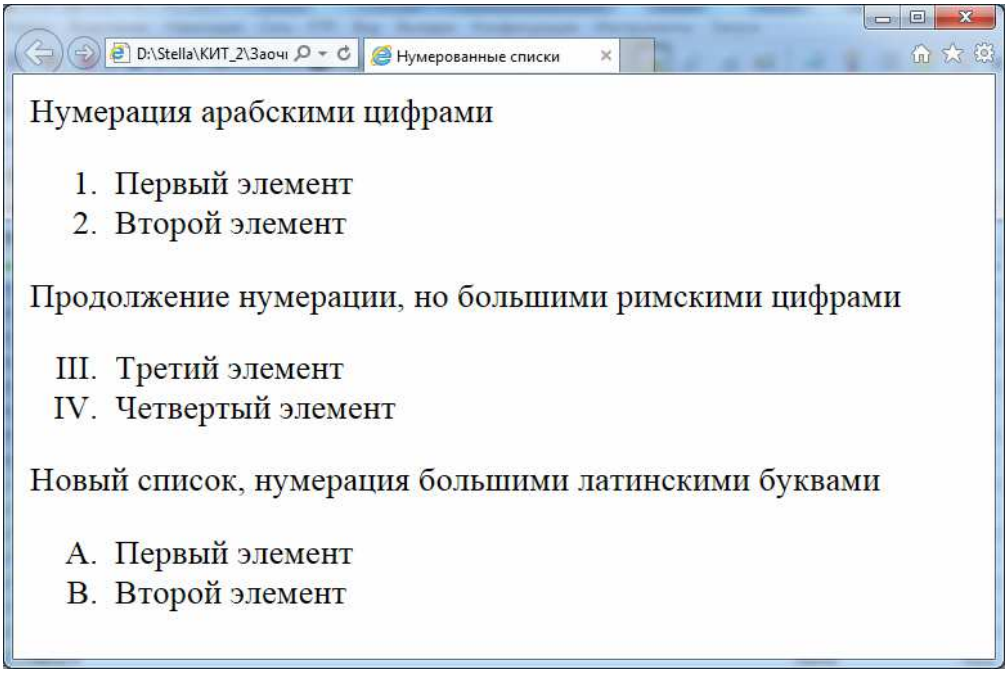


Рис. Л9.3. Результат выполнения задания № 2

Задание № 3. Изменение нумерации списка в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ9_3.HTML или ФАМИЛИЯ9_3.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л9.4).

```
<HTML>
   <HEAD>
      <TITLE> Изменение нумерации списка </TITLE>
   </HEAD>
   <BODY>
      <OL>
      <Li>Первый элемент
      <Li value =10> Второй элемент
      <Li> Третий элемент
      <Li value=20 type=A> Четвертый элемент
      <Li type=A> Пятый элемент
      </OL>
   </BODY>
</HTML>
```

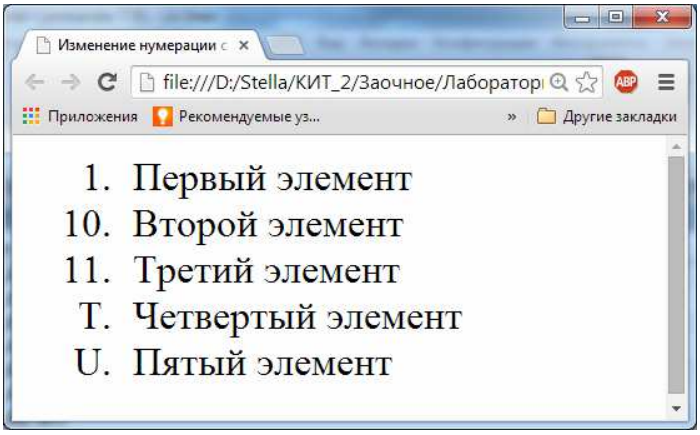


Рис. Л9.4. Результат выполнения задания № 3

Задание № 4. Создание списка определений в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ9_4.HTML или ФАМИЛИЯ9_4.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л9.5).

```
<HTML>
   <HEAD>
      <TITLE> Список определений </TITLE>
```

```
   </HEAD>
   <BODY>
      <DL>
      <DT> HTML (HTML- термин)
         <DD>Широко используемый язык гипертекстовой
         разметки
      <DT>WWW (WWW – термин)
         <DD> От английского Word Wide Web,глобальная
         сеть из соединенных между собой гипертекстовых
         документов
      </DL>
   </BODY>
</HTML>
```

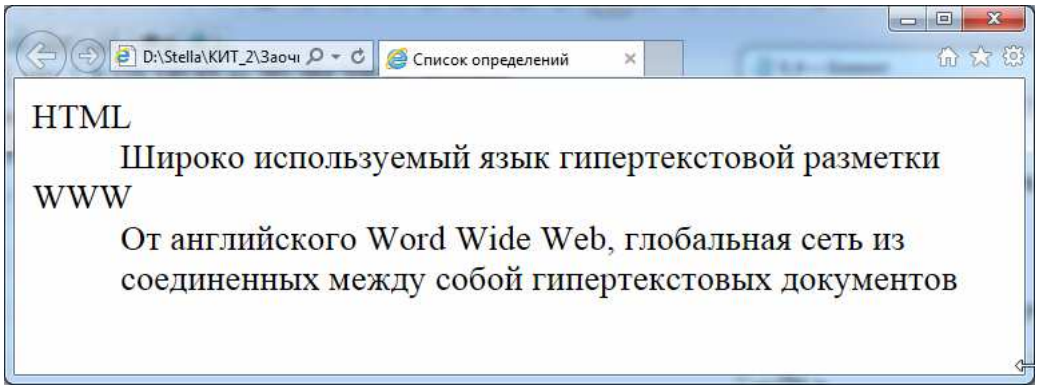


Рис. Л9.5. Результат выполнения задания № 4

Примечание 9.1. Как видно из рисунка, браузер по-разному отображает сами термины и определения этих терминов, причем делает это так, что сразу понятно, где определение, а где термин.

Задание № 5. Создание вложенных списков в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ9_5.HTML или ФАМИЛИЯ9_5.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л9.6).

```
<HTML>
   <HEAD>
      <TITLE> Использование вложенных списков </TITLE>
   </HEAD>
```

```
   <BODY>
      <OL>
      <Li>Первый элемент списка. Имеет две составляющих:
      <UL>
      <Li>Первая составляющая
      <Li>Вторая составляющая
      </UL>
      <Li>Второй элемент списка. Имеет три составляю-
      щих:
      <UL>
      <Li>Первая составляющая. Тоже разделяется на две
      части:
      <UL>
      <Li>Первая часть
      <Li>Вторая часть
      </UL>
      <Li>Вторая составляющая
      </UL>
      <Li>Третий элемент списка.
      </OL>
   </BODY>
</HTML>
```

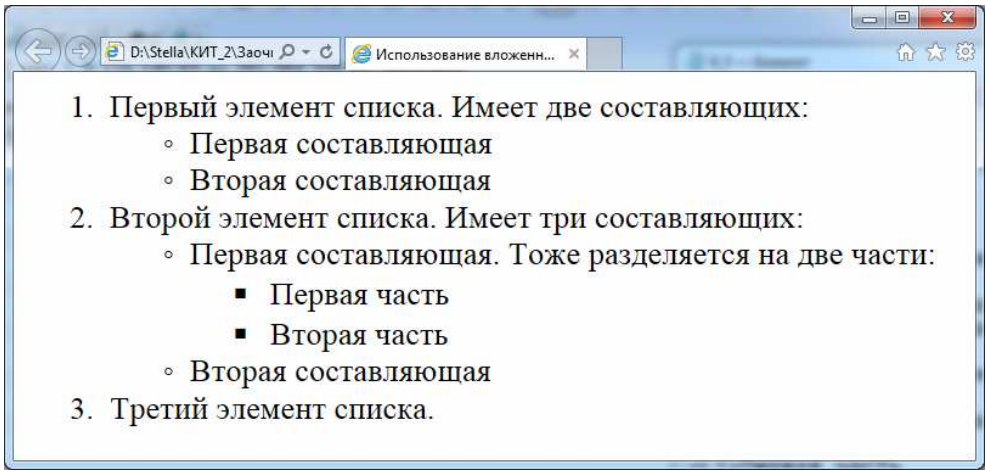


Рис. Л9.6. Результат выполнения задания № 5

Задание № 6. Вложение списков в список определений в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ9_6.HTML или ФАМИЛИЯ9_6.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л9.7).

```
<HTML>
   <HEAD>
      <TITLE> Вложение списков </TITLE>
</HEAD>
   <BODY>
      <DL>
      <DT><STRONG>Неисправность</STRONG>
      <DD>Отсутствие соединения компьютеров сети (PING-
      тест не проходит)
      <DT><STRONG>Возможные причины</STRONG>
      <DD>
      <UL>
      <Li>Отсутствие физического соединения между
      компьютерами
      <UL>
      <Li>Не подключены сетевые провода
      <Li>Не работает активное сетевое оборудование
      <Li>Обрыв сетевых проводов
      <Li>Неисправные сетевые адаптеры
      </UL>
      <Li>Неправильная настройка IP-адресов
      </UL>
      <DT><STRONG>Диагностика</STRONG>
      <DD>
      <OL>
      <Li>Проверить, включены ли сетевые провода в
      адаптеры. Если нет – подключить.
      <Li>Проверить состояние индикации на сетевых
      адаптерах. Если индикаторы горят, то физическое
      соединение компьютеров есть. Нужно проверить IP-
      адреса (см. раздел настройки IP- адресов)
      <Li>Проверить, выключено ли активное сетевое
      оборудование(по состоянию индикации включения
      питания).Выключенное оборудование включить, не-
      исправное заменить.
      <Li>И т. д.
      </OL>
      </DL>
   </BODY>
</HTML>
```

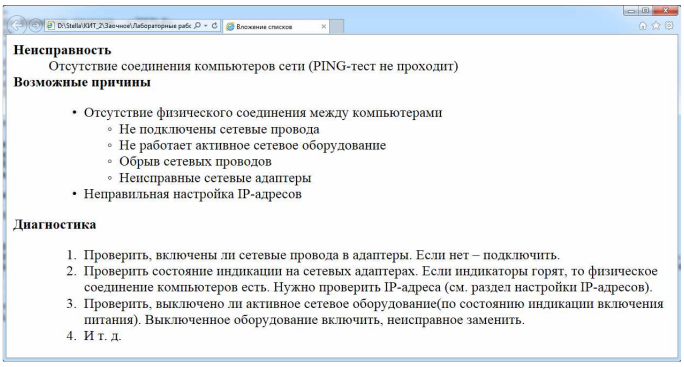


Рис. Л9.7. Результат выполнения задания № 6

Задание № 7. Использование линеек в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ9_7.HTML или ФАМИЛИЯ9_7.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л9.8).

```
<HTML>
<HEAD>
<TITLE> Линейки </TITLE>
</HEAD>
<BODY>
Линейка с настройками по умолчанию
<HR>
Линейки различной толщины
<HR size=4>
<HR size=10>
Плоская линейка
<HR noshade size=10>
Линейки с различным выравниванием
<HR width=50% align=left>
<HR width=50% >
<HR width=50% align=right>
</BODY>
</HTML>
```

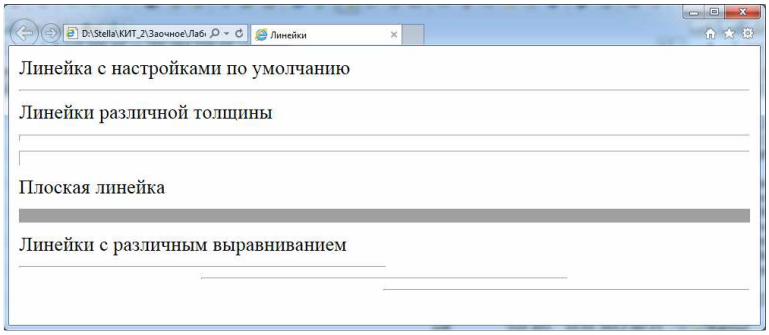


Рис. Л9.8. Результат выполнения задания № 7

ЛАБОРАТОРНАЯ РАБОТА № 10
Таблицы в HTML-документах

Цель выполнения работы: изучение основных приемов создания таблиц в HTML-документах.

Теоретические сведения

Таблицы в HTML представляются множеством ячеек, которые заключены в строки. Совокупность всех строк образует целую таблицу.

Создание таблицы

Для создания таблиц используется четыре пары тегов:

– `<TABLE>…</TABLE >` – устанавливают начало и конец таблицы;

– `<CAPTION >…</CAPTION >` – заголовок таблицы;

– `<TR>…</TR>` – добавляют новую строку в таблицу;

– `<TH>…</TH>` – добавляют в строку ячейку заголовка;

– `<TD>…</TD>` – добавляют в строку обычную ячейку.

Заголовок таблицы

Для каждой таблицы имеется возможность создать заголовок, используя HTML-элемент `CAPTION`, помещенный после `<TABLE>`. Если используется заголовок таблицы, то элемент `CAPTION` должен быть задан непосредственно после тега `<TABLE>` нужной таблицы. Также для любой таблицы может быть создано не более одного заголовка.

Для элемента `CAPTION` можно задать атрибут, определяющий положение заголовка относительно таблицы – `align`. Этот атрибут может принимать следующие значения:

– `top` – заголовок показывается сверху таблицы (используется по умолчанию);

– `bottom` – под таблицей;

– `left` – слева от таблицы;

– `right` – справа от таблицы.

Параметры отображения таблицы

Многие параметры отображения таблицы задаются установкой соответствующих значений атрибутов HTML-элемента `TABLE`. Далее приведен список наиболее используемых атрибутов:

1) align – задает положение таблицы в окне браузера (left, right, center);

2) bgcolor – задает цвет фона таблицы;

3) border – задает толщину внешней границы таблицы (по умолчанию имеет значение 1);

4) bordercolor – цвет границ таблицы (цвет внешней границы и цвет границ ячеек);

5) cellspacing – размер пустого пространства между ячейками таблицы;

6) cellpadding – размер пустого пространства между границами и содержимым ячеек таблицы;

7) frame – задает отображаемые части внешней рамки таблицы, может принимать одно из перечисленных ниже значений:

– void – рамка не отображается (используется по умолчанию),

– above – отображается только верхняя граница,

– below – показывается только нижняя граница,

– hsides – отображаются только верхняя и нижняя границы,

– vsides – показывается правая и левая границы,

– lhs – отображается левая граница,

– rhs – отображается только правая граница,

– box – рамка отображается полностью,

– border – то же самое, что и box;

8) rules – задает, какие именно границы между ячейками должны отображаться, может принимать одно из следующих значений:

– none – границы между ячейками не отображаются (используются по умолчанию),

– group – показываются границы только между группами строк и столбцов,

– rows – отображаются только границы между строками таблицы,

– cols – показываются границы только между столбцами таблицы,

– all – отображаются все границы между ячейками;

9) height – задает рекомендуемую высоту таблицы;

10) width – определяет рекомендуемую ширину таблицы.

В качестве значения атрибутов, задающих размер таблицы, можно использовать как абсолютные значения в пикселях, так и процентные, отображающие долю от ширины окна браузера, занимаемую таблицей.

Параметры отображения строк таблицы

Для настройки особого отображения отдельных строк таблицы используются атрибуты HTML-элемента TR (этот элемент объединяет отдельные ячейки в строки таблицы).

Список основных атрибутов элемента TR:

– align – задает горизонтальное выравнивание текста ячеек строки, может принимать значения left, right, center или justify;

– valign – определяет вертикальное выравнивание текста ячеек строки, может принимать значения top, bottom, middle или baselline;

– bgcolor – задает цвет фона ячеек строки;

– bordercolor – определяет цвет рамки ячеек строки (если рамка отображается);

– height – позволяет указать рекомендуемую высоту ячеек строки;

– width – дает возможность указать рекомендуемую ширину ячеек строки.

Параметры отображения ячеек. Слияние ячеек

Рассмотрим самое эффективное средство настройки отображения данных таблицы – настройку параметров отображения ее отдельных ячеек. Для задания внешнего вида содержимого ячейки используются такие атрибуты HTML-элемента TD:

– align – задает горизонтальное выравнивание текста ячееки, может принимать значения left, right, center или justify;

– valign – определяет вертикальное выравнивание текста ячееки, может принимать значения top, bottom, middle или baselline;

– bgcolor – задает цвет фона ячееки;

– bordercolor – определяет цвет рамки ячееки (если рамка отображается);

– heigt – позволяет указать рекомендуемую высоту ячееки;

– width – дает возможность указать рекомендуемую ширину ячееки;

– colspan – задает количество столбцов для слияния;

– rowspan – опредееляет количество строк для слияния.

Последние два атрибута элемента TD используются для объединения нескольких ячеек при создании более сложных таблиц.

Структурирование таблицы. Группировка строк

HTML поддерживает разбиение таблицы на логически завершенные части: группы строк и столбцов. Для отдельных частей таблицы можно установить общие параметры отображения данных, которые автоматически применяются браузером при отображении таблицы.

В любой таблице, добавляемой в HTML-документ, можно выделить три логически цельные части: *шапка*, *тело таблицы* и *футер* (нижняя часть таблицы, *footer*). Задаются эти части следующими тегами:

– шапка таблицы – <THEAD>;

– тело таблицы – <TBODY>;

– футер (нижняя часть) – <TFOOT>.

Закрывающиеся теги являются необязательными. Внутри этих тегов помещаются строки, относящиеся к соответствующим частям таблицы.

Задание параметров отображения столбцов

При группировке строк таблицы присутсвуют элементы, позволяющие определять столбцы и формировать группы столбцов: <COL> и <COLGROUP>.

HTML-элемент COL задается одиночным тегом <COL>. Он позволяет установить общие параметры отображения всех ячеек, входящих в столбец или столбцы, заданием следующих атрибутов:

– align – задает горизонтальное выравнивание текста ячеек столбца (столбцов), может принимать значения left, right, center или justify;

– valign – определяет вертикальное выравнивание текста ячеек столбца (столбцов), может принимать значения top, bottom, middle или baselline;

– bgcolor – задает цвет фона ячеек столбца (столбцов);

– width – позволяет указать ширину столбца (столбцов);

– span – задает количество столбцов, к которым применяются параметры, заданные в описанных выше атрибутах (по умолчаниию имеет значение 1).

Элемент <COL> не позволяет создавать группы столбцов, но он облегчает настройку внешнего вида таблицы, предоставляет возможность задать настройку одного типа для нескольких столбцов одновременно.

Для группировки используется тег <COLGROUP>.

Задание № 1. Создание простой таблицы в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ10_1.HTML или ФАМИЛИЯ10_1.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис Л10.1).

```
<HTML>
   <HEAD>
      <TITLE> Простейшая таблица</TITLE>
   </HEAD>
   <BODY>
      <TABLE>
         <TR> <TD>1<TD>11<TD>111<TD>1111
         <TR> <TD>2<TD>22<TD>222<TD>2222
         <TR> <TD>3<TD>33<TD>333<TD>3333
         <TR> <TD>4<TD>44<TD>444<TD>4444
      </TABLE>
   </BODY>
</HTML>
```

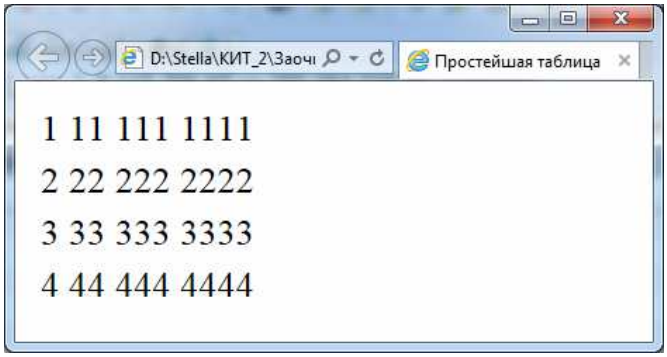


Рис Л10.1. Результат выполнения задания № 1

Задание № 2. Создание простой таблицы с заголовком в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ10_2.HTML или ФАМИЛИЯ10_2.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис Л10.2).

```
<HTML>
   <HEAD>
      <TITLE> Таблица с заголовком </TITLE>
```

```
   </HEAD>
   <BODY>
      <TABLE>
      <CAPTION> <B> Простая таблица, но уже с заголовком
      </B>
      </CAPTION>
         <TR> <TD>1<TD>11<TD>111<TD>1111
         <TR> <TD>2<TD>22<TD>222<TD>2222
         <TR> <TD>3<TD>33<TD>333<TD>3333
         <TR> <TD>4<TD>44<TD>444<TD>4444
      </TABLE>
   </BODY>
</HTML>
```

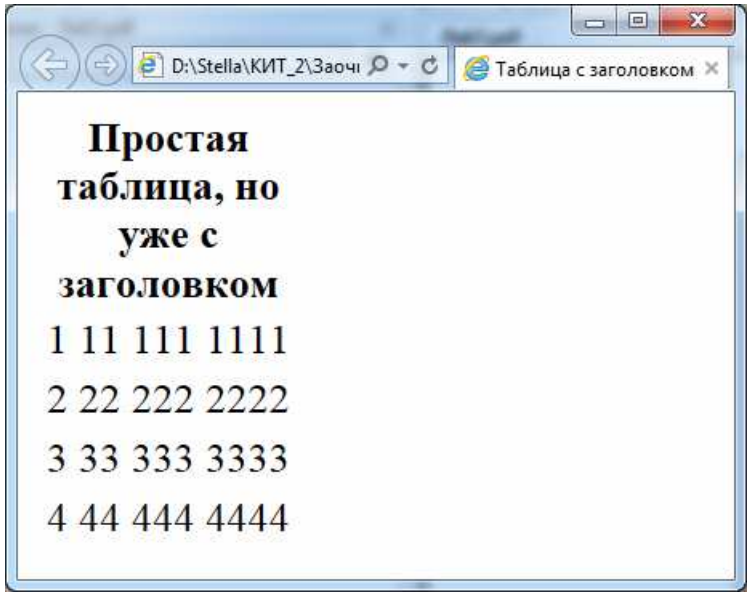


Рис Л10.2. Результат выполнения задания № 2

Задание №3. Оформление строк таблицы с заголовком в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ10_3.HTML или ФАМИЛИЯ10_3.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис Л10.3).

```
<HTML>
   <HEAD>
      <TITLE> Оформление строк таблицы </TITLE>
```

```
  </HEAD>
  <BODY>
    <TABLE align=center border=5 bgcolor=red
    bordercolor = green cellpadding=”5” width=”50%”>
    <CAPTION > <B>Простая таблица, но уже с заголовком
    и оформлением</B> </CAPTION>
      <TR> <TD>1<TD>11<TD>111<TD>1111
      <TR> <TD>2<TD>22<TD>222<TD>2222
      <TR> <TD>3<TD>33<TD>333<TD>3333
      <TR> <TD>4<TD>44<TD>444<TD>4444
    </TABLE>
  </BODY>
</HTML>
```

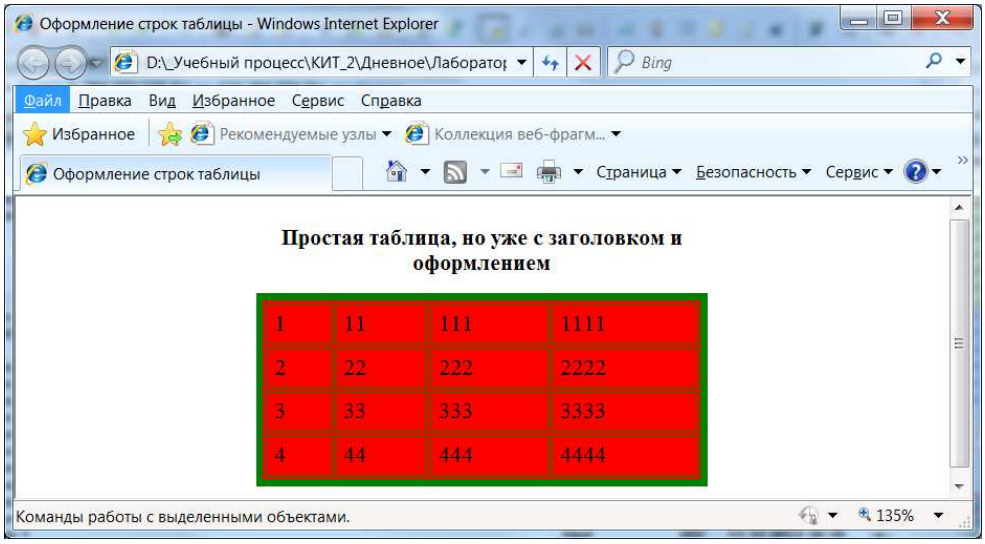


Рис Л10.3. Результат выполнения задания № 3

Задание № 4. Настройка отображения строк таблицы с заголовком в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ10_4.HTML или ФАМИЛИЯ10_4.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л10.4).

```
<HTML>
  <HEAD>
    <TITLE> Настройка отображения строк таблицы
    </TITLE>
  </HEAD>
```

```
   <BODY>
      <TABLE align = center border = 5 bgcolor = red
      bordercolor = green cellpadding = "5"
      width = "50%">
      <CAPTION aling = top> <B> Простая таблица, но уже
      с заголовком и оформлением </B> </CAPTION>
         <TR align=right bgcolor=white> <TD> 1 <TD> 11
         <TD> 111 <TD> 1111
         <TR align = center bordercolor = black> <TD> 2
         <TD> 22 <TD> 222 <TD> 2222
         <TR bgcolor = white> <TD> 3 <TD> 33 <TD> 333
         <TD> 3333
         <TR bordercolor = black> <TD> 4 <TD> 44 <TD>
         444 <TD> 4444
      </TABLE>
   </BODY>
</HTML>
```

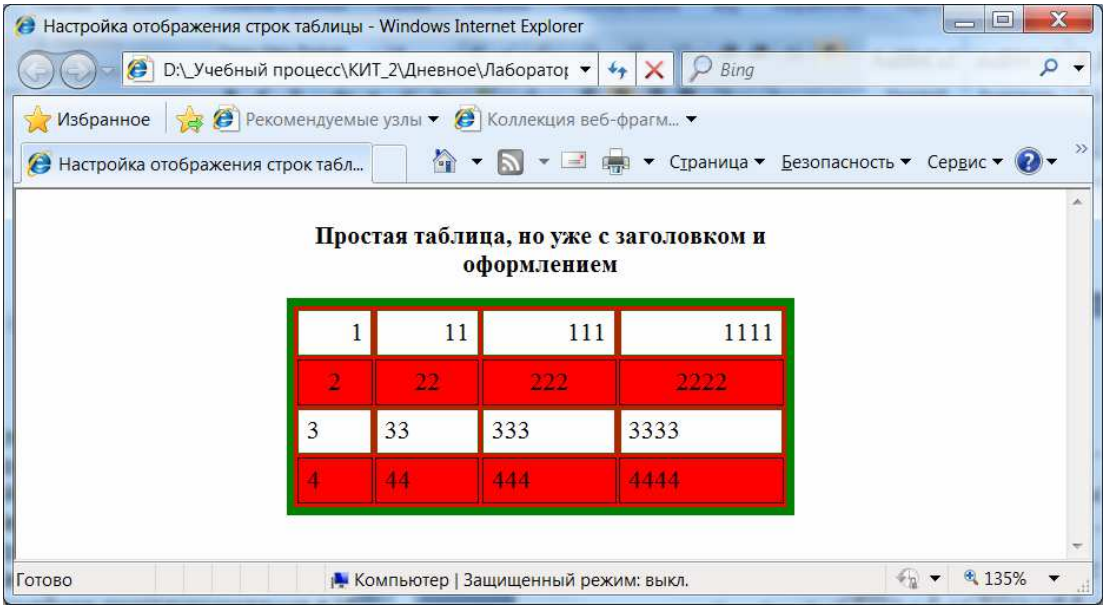


Рис Л10.4. Результат выполнения задания № 4

Задание №5. Создание таблицы с объединенными ячейками в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ10_4.HTML или ФАМИЛИЯ10_4.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л10.5).

```
<HTML>
   <HEAD>
      <TITLE> Таблица с объединенными ячейками
      </TITLE>
   </HEAD>
   <BODY>
      <TABLE align = center border =3 bordercolor =
      green cellpadding = "5" width="50%">
      <CAPTION aling = top> <B> Доходы от продаж за
      второе полугодие 2005 года </B> </CAPTION>
      <!--Формирование первой строки шапки таблицы-->
         <TR align = center>
         <TD rowspan = 2 <B> Филиал\Период </B>
         <TD colspan = 3 <B> 3 квартал </B>
         <TD colspan = 3 <B> 4 квартал </B>
      <!--Формирование второй строки шапки (названия
      месяцев)-->
         <TR align = center>
         <TD> <B> Июль </B> <TD> <B> Август </B> <TD> <B>
         Сентябрь </B >
         <TD> <B> Октябрь </B> <TD> <B> Ноябрь </B>
         <TD> <B> Декабрь </B>
      <!—Дале следуют строки с данными (первая ячейка
      каждой строки – название филиала -->
         <TR align=right><TD align=left> Филиал 1 <TD>
         123123 <TD> 323233 <TD>323453<TD> 231423 <TD>
         323212<TD> 243673
         <TR align=right><TD align=left> Филиал 2 <TD>
         223523 <TD> 225243 <TD>31423<TD> 212445 <TD>
         373812<TD> 274673
         <TR align=right><TD align=left> Филиал 3 <TD>
         183123 <TD> 323453 <TD>323453<TD> 231423 <TD>
         323212<TD> 243673
         <TR align=right><TD align=left> Филиал 4 <TD>
         123123 <TD> 323233 <TD>323453<TD> 231423 <TD>
         323212<TD> 243673
      </TABLE>
   </BODY>
</HTML>
```

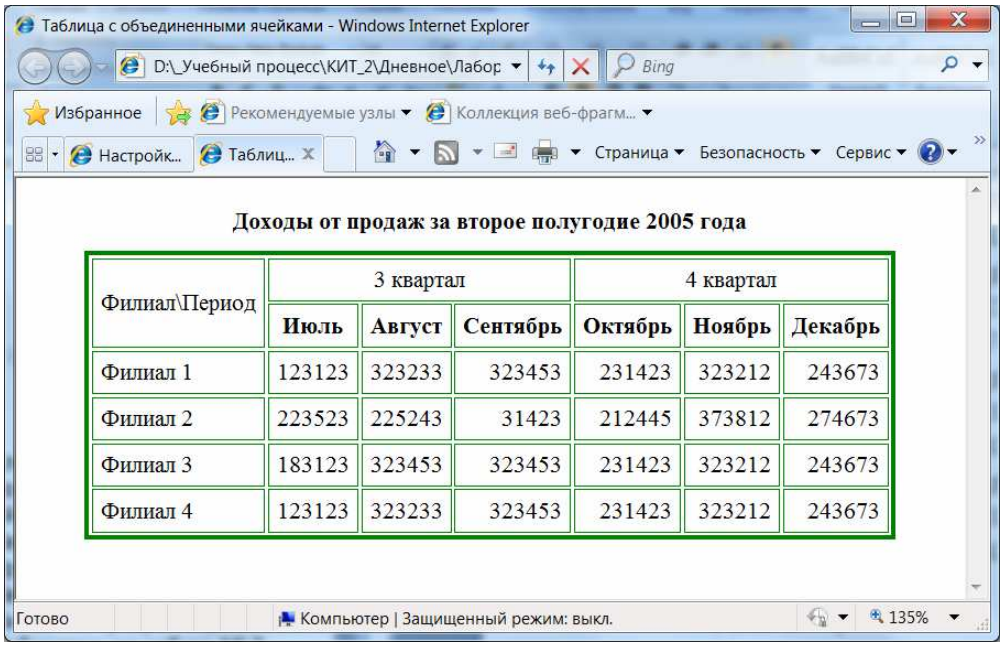


Рис Л10.5. Результат выполнения задания № 5

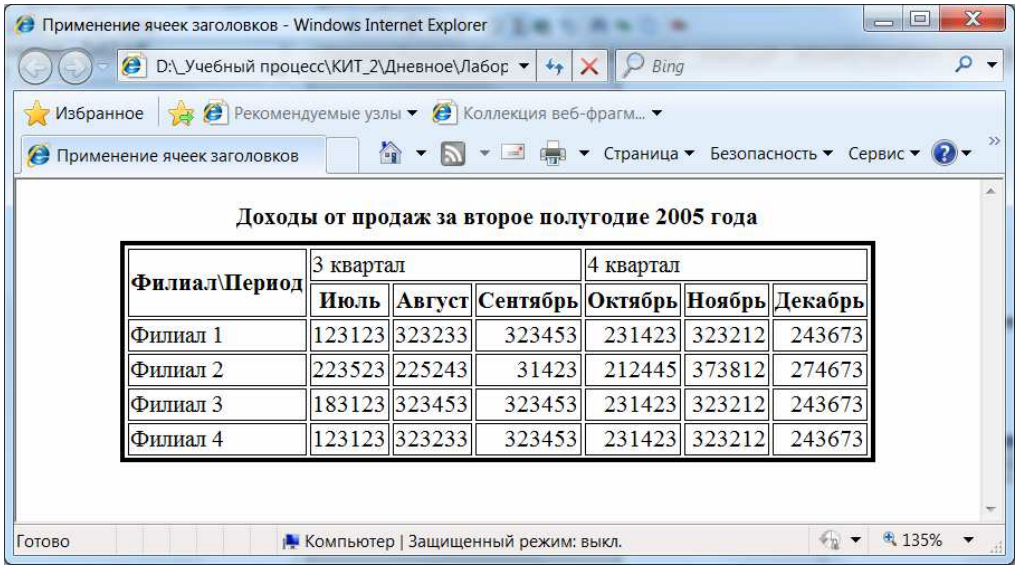
Задание № 6. Создание таблицы с заголовочными ячейками в HTML-документе

В предыдущем примере для визуального выделения ячеек заголовков таблицы использовалось форматирование текста внутри HTML-элемента TD вручную, этот способ не очень удобен. Использование тега HTML-элемента <TH> и </TH> позволит лучше структурировать содержимое HTML-документа.

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ10_6.HTML или ФАМИЛИЯ10_6.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л10.6).

```
<HTML>
  <HEAD>
    <TITLE> Применение ячеек заголовков </TITLE>
  </HEAD>
  <BODY>
    TABLE align=center border=3 bordercolor=black>
    <CAPTION aling=top><B>Доходы от продаж за второе
    полугодие 2005 года</B></CAPTION>
    <!--Формирование первой строки шапки таблицы-->
      <TR>
      <TH rowspan=2>Филиал\Период
```

```
      <TD colspan=3 > 3 квартал
      <TD colspan=3 > 4 квартал
   <!--Формирование второй строки шапки (названия
   месяцев)-->
      <TR>
      <TH>Июль<TH>Август<TH>Сентябрь
      <TH>Октябрь<TH>Ноябрь<TH>Декабрь
   <!—Далее следуют строки с данными (первая ячейка
   каждой строки – название филиала) -->
      <TR align=right><TD align=left> Филиал 1 <TD>
      123123 <TD> 323233 <TD>323453<TD> 231423 <TD>
      323212<TD> 243673
      <TR align=right><TD align=left> Филиал 2 <TD>
      223523 <TD> 225243 <TD>31423<TD> 212445 <TD>
      373812<TD> 274673
      <TR align=right><TD align=left> Филиал 3 <TD>
      183123 <TD> 323453 <TD>323453<TD> 231423 <TD>
      323212<TD> 243673
      <TR align=right><TD align=left> Филиал 4 <TD>
      123123 <TD> 323233 <TD>323453<TD> 231423 <TD>
      323212<TD> 243673
   </TABLE>
  </BODY>
</HTML>
```



Доходы от продаж за второе полугодие 2005 года

Филиал\Период	3 квартал			4 квартал		
	Июль	Август	Сентябрь	Октябрь	Ноябрь	Декабрь
Филиал 1	123123	323233	323453	231423	323212	243673
Филиал 2	223523	225243	31423	212445	373812	274673
Филиал 3	183123	323453	323453	231423	323212	243673
Филиал 4	123123	323233	323453	231423	323212	243673

Рис Л10.6. Результат выполнения задания № 6

Задание № 7. Группировка строк таблицы в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ10_7.HTML или ФАМИЛИЯ10_7.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л10.7).

```
<HTML>
   <HEAD>
      <TITLE> Группировка строк таблицы </TITLE>
   </HEAD>
   <BODY>
      <TABLE align = center border = 3
      bordercolor = black rules = groups>
      <CAPTION aling = top> <B> Доходы от продаж за вто-
      рое полугодие 2005 года </B> </CAPTION>
      <THEAD>
      <!--Формирование первой строки шапки таблицы-->
         <TR>
         <TH rowspan = 2> Филиал\Период
         <TD colspan=3> 3 квартал
         <TD colspan=3> 4 квартал
      <!--Формирование второй строки шапки (названия ме-
      сяцев)-->
         <TR>
         <TH> Июль <TH> Август <TH> Сентябрь
         <TH> Октябрь <TH> Ноябрь <TH> Декабрь
         <TBODY align = right>
      <!—Далее следуют строки с данными (первая ячейка
      каждой строки – название филиала) -->
         <TR> <TD align = left> Филиал 1 <TD> 123123 <TD>
         323233 <TD> 323453 <TD> 231423 <TD> 323212 <TD>
         243673
         <TR> <TD align = left> Филиал 2 <TD> 223523 <TD>
         225243 <TD> 31423 <TD> 212445 <TD> 373812 <TD>
         274673
         <TR> <TD align = left> Филиал 3 <TD> 183123 <TD>
         323453 <TD> 323453 <TD> 231423 <TD> 323212<TD>
         243673
         <TR> <TD align = left> Филиал 4 <TD> 123123 <TD>
         323233 <TD> 323453 <TD> 231423 <TD> 323212 <TD>
         243673
         <TBODY align = right>
      <!—Стока с итоговыми данными-->
         <TR><TD align = left> Всего: <TD> 123123 <TD>
         323233 <TD>323453<TD> 231423 <TD> 323212<TD>
         243673
      </TABLE>
   </BODY>
</HTML>
```

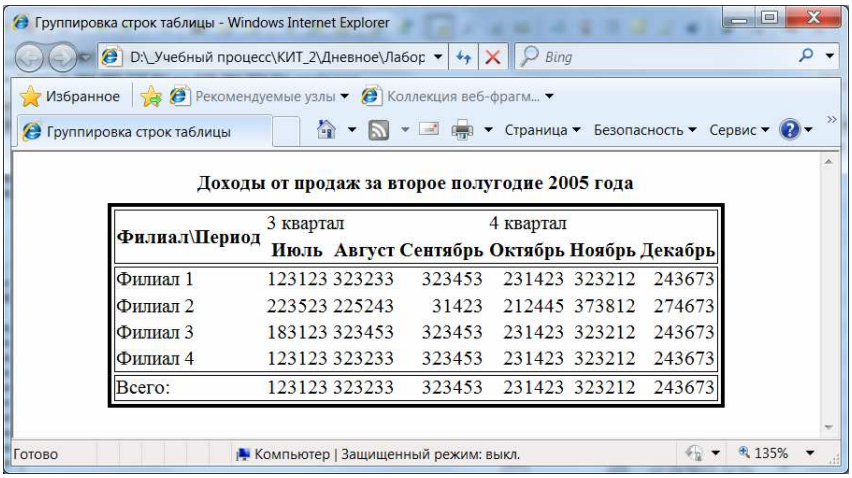


Рис Л10.7. Результат выполнения задания № 7

Чтобы подчеркнуть отделение частей таблицы друг от друга, в этой таблице задано отображение границ только между группами строк и столбцов (см. атрибут `rules` элемента `TABLE`). В таблице сгуппированы только строки, поэтому и отображены только горизонтальные границы.

Задание №8. Задание параметров отображения столбцов таблицы в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ10_8.HTML или ФАМИЛИЯ10_8.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л10.8).

```
<HTML>
   <HEAD>
      <TITLE> Использование элемента COL </TITLE>
   </HEAD>
   <BODY>
      <TABLE align = center border=3
      bordercolor = black rules=groups>
      <CAPTION aling = top> <B> Доходы от продаж за
      второе полугодие 2005 года</B></CAPTION>
      <THEAD>
      <!--Формирование первой строки шапки таблицы-->
         <TR bgcolor=magenta>
         <TH rowspan=2>Филиал\Период
         <TD colspan=3> 3 квартал
         <TD colspan=3> 4 квартал
      <!--Формирование второй строки шапки (названия
      месяцев)-->
```

```
        <TR bgcolor=magenta>
        <TH> Июль <TH> Август <TH> Сентябрь
        <TH> Октябрь <TH> Ноябрь <TH> Декабрь
        <TBODY align = right>
    <!--Определение столбцов таблицы-- >
      <COL align = left bgcolor = green>
      <COL span = 3 bgcolor = blue>
      <COL span = 3 bgcolor = yellow>
    <!—Дале следуют строки с данными (первая ячейка
    каждой строки – название филиала) -->
      <TR> <TD align = left> Филиал 1 <TD> 123123
      <TD> 323233 <TD> 323453 <TD> 231423 <TD>
      323212 <TD> 243673
      <TR > <TD align = left> Филиал 2 <TD> 223523
      <TD> 225243 <TD> 31423 <TD> 212445 <TD>
      373812<TD> 274673
      <TR > <TD align = left> Филиал 3 <TD> 183123
      <TD> 323453 <TD> 323453 <TD> 231423 <TD>
      323212 <TD> 243673
      <TR> <TD align = left> Филиал 4 <TD> 123123
      <TD> 323233 <TD> 323453 <TD> 231423 <TD>
      323212 <TD> 243673
      <TBODY align = right>
    <!—Стока с итоговыми данными-->
        <TR bgcolor = red> <TD> Всего: <TD> 123123
      <TD> 323233 <TD> 323453 <TD> 231423 <TD>
      323212 <TD> 243673
    </TABLE>
  </BODY>
</HTML>
```

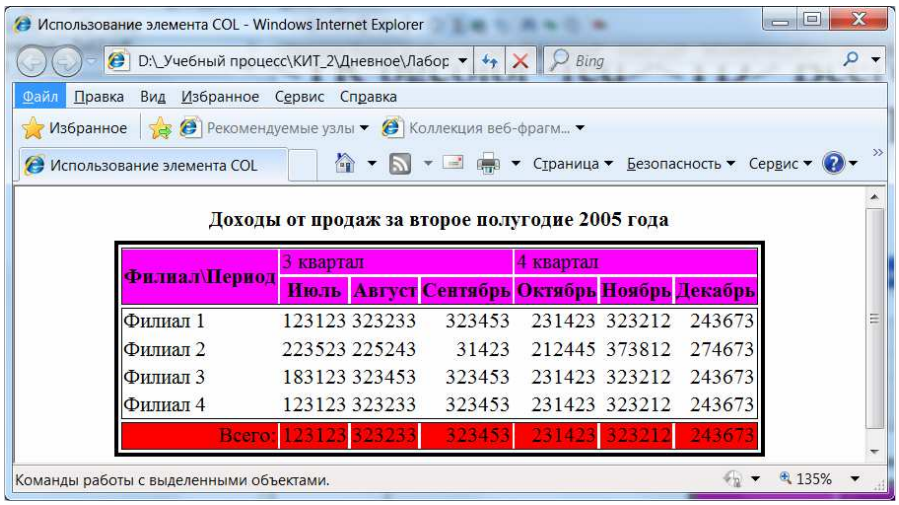


Рис Л10.8. Результат выполнения задания № 8

ЛАБОРАТОРНАЯ РАБОТА № 11
Формы в HTML-документах

Цель выполнения работы: изучение основных приемов создания форм в HTML-документах.

Теоретические сведения

Формы – замечательная возможность HTML, позволяющая с привлечением довольно небольшого количества усилий организовать взаимодействие с пользователями программ, работающих на удаленных серверах Сети.

При заполнении анкеты на сайтах, например, при регистрации ящика электронной почты, данные вводятся прямо в окне браузера. После ввода данных нажимается кнопка Отправить. Далее браузер упаковывает и отсылает введенные данные на сервер, где их обрабатывает специализированное приложение (CGI-приложение). Формы нужны для того, чтобы можно было организовать ввод данных от пользователя.

HTML-документы с формами отличаются от обычных документов наличием различных элементов управления: полей ввода текста, флажков, кнопок и др. Данные, введенные в форму, в некоторых случаях могут обрабатываться и на стороне клиента с помощью сценариев. Мы рассмотрим создание и настройку форм, данные которых предназначены для отправки CGI (Common Gateway Interface)-приложению.

Создание формы

Для вставки формы в HTML-документ используется элемент FORM. Он задается парными тегами <FORM> и </FORM>. Между этими тегами помещаются описания элементов управления формы. Здесь может быть помещен и другой текст с использованием разметки средствами HTML. Этот текст обычно используется для пояснения, какие данные и в какой элемент управления нужно вводить.

При создании формы используются следующие атрибуты элемента FORM:

action – обязательный для каждой формы параметр, URI программы-обработчика данных форм;

method – задает способ отправки данных, введенных в форму, может принимать значения get (используется по умолчанию) или post;

enctype – задает тип данных формы, если используется метод отправки post; по умолчанию имеет значение application/x-www-form-urlencoded; при необходимости передачи файлов используется значение multipart/form-data;

accept-charset – применяется при передаче файлов, позволяет указать, какие кодировки используются для каждого из файлов (список строковых значений – названий кодировок), по умолчанию устанавливается значение UNKNOWN (приложение на сервере должно само определять кодировки);

accept – описывает типы (MIME-типы), передаваемые серверу; если этот параметр не использовать, то серверное приложение должно уметь само определять типы передаваемых ему файлов.

Методы отправки данных форм и их отличия

Метод get используется при отправке небольших объемов данных, для которых достаточно набора символов кодировки ASCII.

Метод post используется при отправке практически любых данных.

Стандартные элементы управления

Стандартными являются все элементы управления, которые можно поместить на HTML-форму. Просто нужно объединить и назвать элементы управления, которые используются чаще всего: однострочное текстовое поле, поле для ввода пароля, флажки, переключатели, кнопки (как пользовательские, так и выполняющие стандартные действия), поля имен файлов.

Все упомянутые элементы управления отображаются браузером. Для обозначения всех этих элементов управления используется один HTML-элемент – INPUT. Этот элемент задается одиночным тегом <INPUT> и имеет следующие атрибуты:

– type – принимает строку, задающую тип элемента управления (по умолчанию используется строка text и создается соответственно поле для ввода текста), возможные значения и специфика работы соответствующих элементов управления рассмотрены далее;

– name – используется для задания имени элементу управления (строка, которая помимо идентификации элемента управления добавляется в данные, отсылаемые серверу);

– value – начальное значение для полей ввода текста и полей для указания имен файлов, так же используется как надпись таких элементов

управления, как кнопки; необязательно для всех элементов управления, кроме флажков и переключателей;

– `checked` – булев атрибут, если он установлен, то флажок или переключатель считается (и отображается браузером) установленным;

– `disabled` – булев атрибут, установка которого не позволяет пользователю работать с элементом управления;

– `readonly` – булев атрибут, позволяет запретить изменение состояния элемента управления (работает только для текстовых полей и поля выбора файла, так что для остальных элементов управления лучше использовать атрибут `disabled`, однако при этом данные деактивированных элементов управления не отправляется серверу);

– `size` – задает размер элемента управления (единицы измерения и действие специфичны для разных элементов управления);

– `maxlenth` – задает максимальную длину текста, который может быть введен в текстовые поля (положительное численное значение);

– `src` – для элемента управления `image` задает расположение используемого изображения;

– `title` – описание элемента управления (может отображаться браузерами как всплывающая подсказка);

– `align` – задает горизонтальное выравнивание элемента управления, работает так же, как и для любого другого HTML-элемента, поддерживающего этот атрибут;

– `tabindex` – номер элемента управления при навигации при помощи табуляции;

– `accesskey` – «горячая» клавиша для элемента управления (для перехода к элементу управления нужно нажать Alt и заданную клавишу).

Таблица Л11.1

Значения атрибута type

Значение атрибута	Особенности
`text`	Создается однострочное текстовое поле. Значение атрибута `value` отображается в качестве текста по умолчанию. Атрибут `size` воспринимается как количество символов, которое может отображаться в поле без необходимости горизонтальной прокрутки текста. Атрибут `maxlength` задает максимальное количество символов, которые могут быть введены в поле.

Окончание таблицы Л11.1

password	Действие этого значения аналогично действию значения `text`, но дополнительно скрывается вводимый пользователем текст (заменяется символом «*» или другим). Используется для ввода конфиденциальной информации типа паролей.
checkbox	Создает элемент управления флажок. Задавать значение атрибута `value` нужно обязательно, т.к. именно это значение отправляется серверу, если флажок установлен. Можно использовать несколько элементов управления `checkbox` с одинаковым значением атрибута `name` для обеспечения возможности задания нескольких значений одного свойства.
radio	Создает поле для ввода имени файла с возможностью выбора файла с помощью диалогового окна открытия файла или без него, что зависит от браузера. Содержимое выбранного файла или файла пересылки вместе с формой. Для корректной работы необходима установка значения атрибута `method` формы в `post`, а `enctype` – в `multipart/form-data`.Значение атрибута `value` используется как имя файла по умолчанию. Значение атрибута `size` задается и работает аналогично элементу управления `text`.
submit	Создает кнопку, при нажатии которой браузер отправит форму. Значение атрибута `value` задает надпись на кнопке.
image	Создает изображение, при щелчке на котором произойдет отправка формы браузером. При этом на сервер передаются также координаты указателя мыши относительно левого верхнего угла изображения. Атрибут `scr` задает расположение используемого изображения.
reset	Создает кнопку, при нажатии которой значения всех элементов управления будут заменены значениями по умолчанию. Значение `value` задает надпись на кнопке.
button	Создает пользовательскую кнопку. При нажатии или другом действии с кнопкой может выполняться ассоциированный сценарий. Значение атрибута `value` используется как надпись на кнопке.
hidden	Создает скрытый элемент управления. Для него задаются атрибуты `value` и `name`. Пользователь не видит и не может изменить содержимое этого элемента управления. Однако данные скрытого элемента управления отправляются серверу вместе с остальными данными формы
file	Создает поле для ввода имени файла с возможностью выбора файла с помощью диалогового окна открытия файла или без него, что зависит от браузера. Содержимое выбранного файла или файла пересылки вместе с формой. Для корректной работы необходима установка значения атрибута `method` формы в `post`, а enctype – в `multipart/form-data`. Значение атрибута `value` используется как имя файла по умолчанию. Значение атрибута `size` задается и работает аналогично элементу управления `text`.

Многострочное текстовое поле

Для редактирования текста в форму можно вставить элемент управления – многострочное текстовое поле. Обозначается этот элемент TEXTAREA и задается парными тегами. Отличие этого элемента управления от обычного текстового поля – начальное значение не задается атрибутом value, а записывается между тегами <TEXTAREA> и </TEXTAREA>.

При отображении в текстовом поле форматирование текста сохраняется как при использовании тега PRE. Размер элемента управления задается не атрибутом size, а с помощью следующих атрибутов:

– rows – задает количество строк, которые отображаются без необходимости вертикальной прокрутки;

– cols – задает максимальное количество символов в строке текстового поля, отображаемое без переноса.

Создание кнопки BUTTON

В HTML предусмотрен отдельный элемент BUTTON на случай, если разработчику формы не хватит возможностей обычных командных кнопок.

При помощи BUTTON создаются кнопки, на которые можно поместить практически все, за исключением других форм, фреймов и т.д.

Используется этот вид кнопок аналогично кнопкам, создаваемым HTML-элементом INPUT со значением атрибута type, равным button.

Отличаются кнопки button прежде всего созданием: для задания используются парные теги <BUTTON> и </BUTTON>, между которыми помещается форматированный текст.

Меню

При создании форм можно использовать еще один достаточно удобный элемент управления, позволяющий выбрать одну или несколько альтернатив из списка – меню (или список параметров). Добавляется в форму этот элемент управления использованием трех HTML-элементов: SELECT, OPTGROUP и OPTION, хотя обязательно использовать только первый и третий.

Элемент SELECT является контейнером пунктов меню, задается при помощи парных тегов <SELECT> и </SELECT> и поддерживает атрибуты:

name – имя меню, которое также используется как название параметра при отправке формы;

size – задает количество строк, одновременно видимых пользователем;

multiple – булев атрибут, позволяющий разрешить выделение нескольких пунктов одновременно;

disabled – булев атрибут, деактивирует меню;

title – текст подсказки для меню;

tabindex – номер при перемещении между элементами управления при помощи табуляции.

Элемент управления меню может представляться различными браузерами по-разному: в виде списка, в виде раскрывающегося списка, в виде раскрывающегося меню.

Элемент OPTION используется для задания отдельных пунктов меню. Задается парными тегами <OPTION> и </OPTION>, между которыми помещается текст пунктов меню. Имеет атрибуты:

– value – текст, который будет отправлен с формой при выборе пункта меню;

– selected – булев атрибут, позволяющий выделить пункт меню по умолчанию (не следует устанавливать для нескольких пунктов одного меню, если не установлен атрибут multiple);

– disabled – булев атрибут, запрещает выбор пункта меню (правда, в реализации меню списками это сделать сложно).

Третий элемент OPTGROUP используется для группировки пунктов меню. Создаются группы заключением HTML-элементов OPTION, определяющих пункты меню одной группы, в парные теги <OPTGROUP> и </OPTGROUP>. Настроить параметры группы можно с использованием атрибутов:

– label – строка с подписью для группы;

– disabled – запретить выбор пунктов меню, принадлежащих этой группе (аналогично атрибуту disabled для элемента OPTION).

Подписи элементов управления

Некоторые элементы управления, например кнопки, содержат надписи, по которым пользователь может определить назначения этих элементов управления. А для других элементов типа текстовых полей, флажков и т.п. приходится создавать подписи вручную, смешивая определения элементов управления формы с текстом.

Существует еще один способ задания подписей к элементам управления – использование HTML-элемента LABEL, но задание подписей осуществить таким способом сложнее.

Группировка элементов управления

При создании форм есть возможность сделать более выразительным общее предназначение некоторых элементов управления, дополнительно заключив их в рамку так, как показано в задании № 6. Рамку можно подписать, а можно оставить без подписи.

Создается рамка при помощи HTML-элемента FIELDSET. Между парными тегами <FIELDSET> и </FIELDSET> помещаются определения элементов управления, принадлежащих группе. Вокруг этих элементов управления и будет нарисована рамка.

Для создания подписи к рамке нужно внутри элемента FIELDSET определить элемент LEGEND. Задается этот HTML-элемент при помощи парных тегов <LEGEND> и </LEGEND>, между которыми помещается текст подписи. Можно задать выравнивание подписи при помощи атрибута align, однако следует знать, что разные браузеры по-разному реализуют значения этого атрибута, а некоторые значения и совсем не поддерживают.

Задание № 1. Примеры элементов управления, используемых в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ11_1.HTML или ФАМИЛИЯ11_1.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л11.1).

```
<HTML>
  <HEAD>
    <TITLE> Типы элементов управления </TITLE>
  </HEAD>
  <BODY>
    <FORM action="…" method="post"
    enctype="multipart/form-data">
      Текстовое поле <INPUT type="text"
      name = "type_text"> <BR>
```

```
        Поле для ввода пароля: <INPUT type="password"
        name="type_password" alt="asfdsvdf"> <BR>
        Флажок: <INPUT type="checkbox"
        name="type_check" value="chk1" checked><BR>
        Для переключателя: <INPUT type="radio" name=
        "type_radio" value="1" checked> и <INPUT
        type="radio" name= "type_radio" value="2"
        alt="asfdsvdf">
        <BR>
        Поле с именами файлов: <INPUT type="file"
        name= "type_file" ><BR>
        <INPUT type="submit" value= Отправка данных><BR>
        Щелчок на этом изображении также приведет к
        отправке данных:
        <INPUT type="image" scr="submit.gif"
        alt="asfdsvdf">
        <BR>
        <INPUT type="reset" value="Сброс значений по-
        лей"> <BR>
        <INPUT type="button" value="Пользовательская
        кнопка" alt="asfdsvdf">
    </FORM>
  </BODY>
</HTML>
```

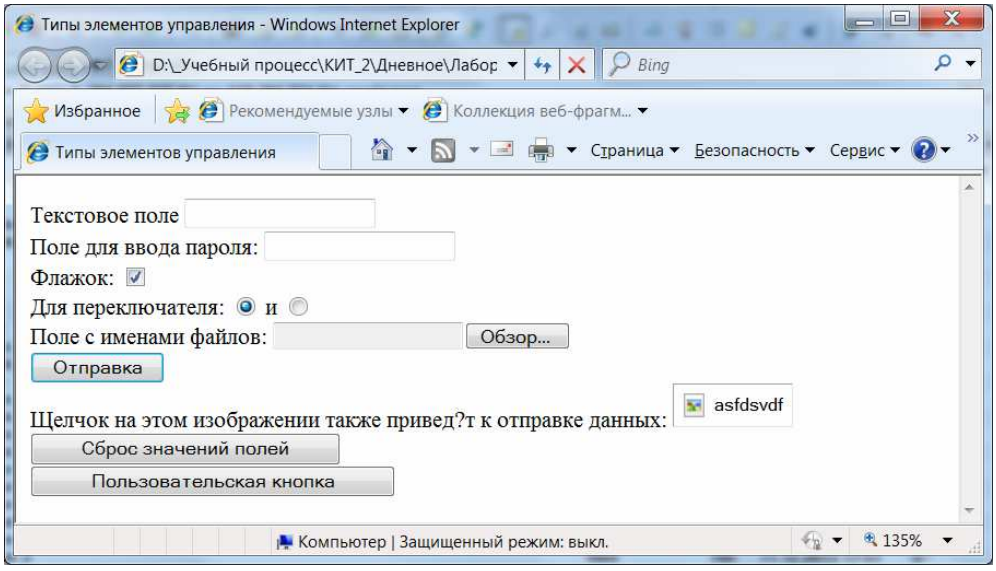


Рис. Л11.1. Результат выполнения задания № 1

Задание № 2. Многострочное текстовое поле в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ11_2.HTML или ФАМИЛИЯ11_2.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л11.2).

```
<HTML>
   <HEAD>
      <TITLE> Многострочное текстовое поле </TITLE>
   </HEAD>
   <BODY>
      <TEXTAREA name="txtText" rows=10 cols=40>
         Первый перевод строки не учитывается.
         Все последующие переводы
         строки учитываются
         Учитываются также отступы от начала строки
         Можно использовать зарезервированные символы
         типа<> & и прочие
      </TEXTAREA>
   </BODY>
</HTML>
```

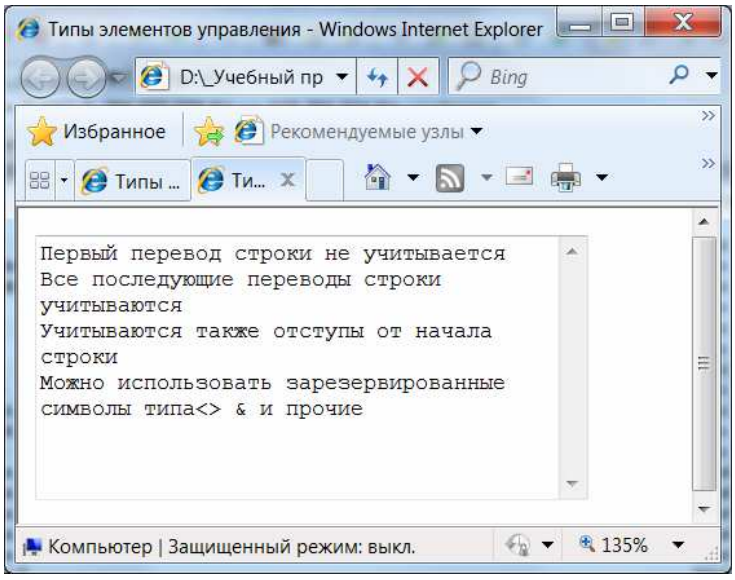


Рис. Л11.2. Результат выполнения задания № 2

Задание №3. Создание кнопки BUTTON в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ11_3.HTML или ФАМИЛИЯ11_3.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л11.3).

```
<HTML>
   <HEAD>
      <TITLE> Создание кнопки </TITLE>
   </HEAD>
   <BODY>
      <BUTTON name="cmbSomeButton" type="button">
         <BR>
      <TABLE border=1>
      <CAPTION> Таблица</CAPTION>
         <TR><TD>1<TD>2
         <TR><TD>3<TD>4
      </TABLE>
         <BR>
      </BUTTON>
   </BODY>
</HTML>
```

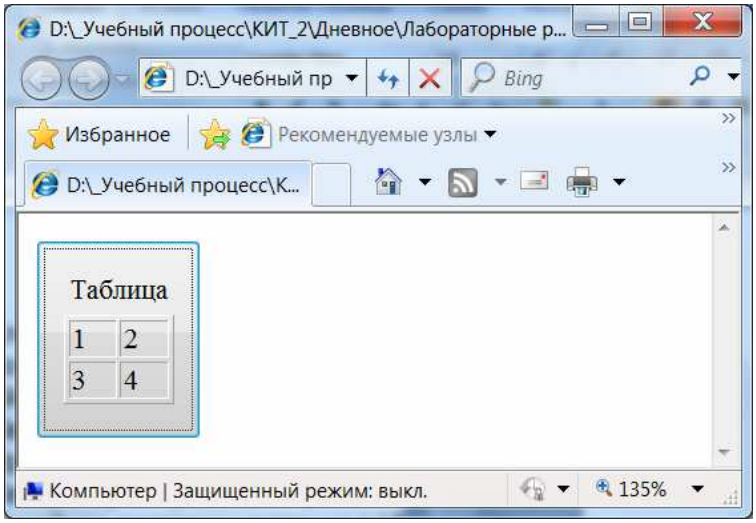


Рис Л11.3. Результат выполнения задания № 3

Задание №4. Группировка строк таблицы в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ11_4.HTML или ФАМИЛИЯ11_4.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л11.4).

```
<HTML>
   <HEAD>
      <TITLE> Создание простых меню </TITLE>
   </HEAD>
   <BODY>
```

```
      size=1
      <SELECT name = "menu1">
         <OPTION value = "1.1"> Пункт 1
         <OPTION value = "1.2"> Пункт 2
         <OPTION value = "1.3"> Пункт 3
         <OPTION value = "1.4"> Пункт 4
         <OPTION value = "1.5"> Пункт 5
         <OPTION value = "1.6"> Пункт 6
      </SELECT>
      size=4
      <SELECT name = "menu2" size="4">
         <OPTION value = "2.1"> Пункт 1
         <OPTION value = "2.2"> Пункт 2
         <OPTION value = "2.3"> Пункт 3
         <OPTION value = "2.4"> Пункт 4
         <OPTION value = "2.5"> Пункт 5
         <OPTION value = "2.6"> Пункт 6
      </SELECT>
   </BODY>
</HTML>
```

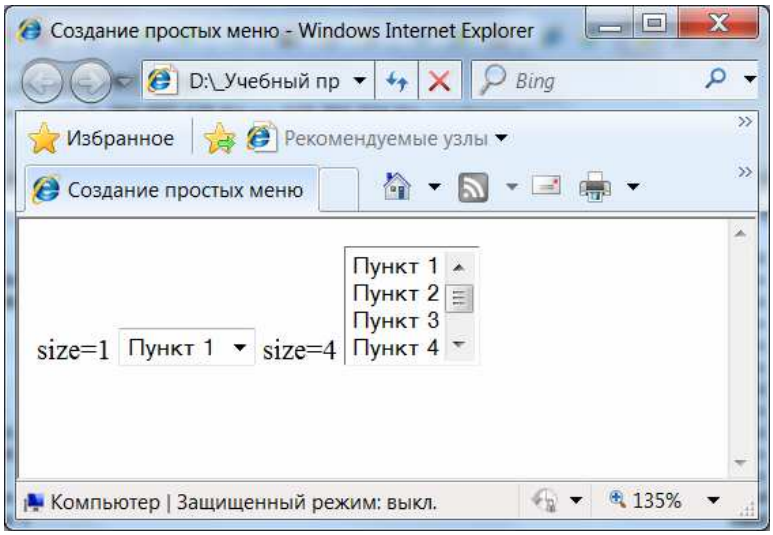


Рис. Л11.4. Результат выполнения задания № 4

Задание №5. Разбиение пунктов меню на группы для двух меню в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ11_5.HTML или ФАМИЛИЯ11_5.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л11.5).

```
<HTML>
   <HEAD>
      <TITLE> Разбиение пунктов меню </TITLE>
   </HEAD>
   <BODY>
      size=1
      <SELECT name = "menu1">
      <OPTGROUP label = "Группа 1">
         <OPTION value = "1.1.1"> Пункт 1.1
         <OPTION value = "1.1.2"> Пункт 1.2
         <OPTION value = "1.1.3"> Пункт 1.3
      </OPTGROUP>
      <OPTGROUP label = "Группа 2">
         <OPTION value = "1.2.1"> Пункт 2.1
         <OPTION value = "1.2.2"> Пункт 2.2
         <OPTION value = "1.2.3"> Пункт 2.3
      </OPTGROUP>
      </SELECT>
      size=4
      <SELECT name = "menu2" size="4">
      <OPTGROUP label = "Группа 1">
         <OPTION value = "2.1.1"> Пункт 1.1
         <OPTION value = "2.1.2"> Пункт 1.2
         <OPTION value = "2.1.3"> Пункт 1.3
      </OPTGROUP>
      <OPTGROUP label = "Группа 2">
         <OPTION value = "2.2.1"> Пункт 2.1
         <OPTION value = "2.2.2"> Пункт 2.2
         <OPTION value = "2.2.3"> Пункт 2.3
      </OPTGROUP>
      </SELECT>
   </BODY>
</HTML>
```

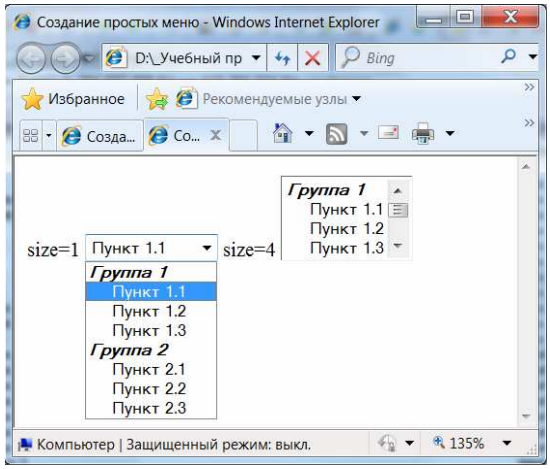


Рис. Л11.5. Результат выполнения задания № 5

Задание № 6. Группировка элементов управления в HTML-документе

Набрать в *Блокноте* приведенный ниже пример. Сохранить как ФАМИЛИЯ11_6.HTML или ФАМИЛИЯ11_6.HTM. Открыть его в браузере Internet Explorer и просмотреть результат (рис. Л11.6).

```
<HTML>
   <HEAD>
      <TITLE> Группировка элементов управления
</TITLE>
   </HEAD>
   <BODY>
      <FORM action ="somesite.com/cgi-bin/proc.exe">
      <H1>Регистрация почтового ящика</H1>
      <P>
      <FIELDSET>
      <LEGEND>Персональная информация</LEGEND>
         Фамилия:<INPUT name= "personal_lastname" type=
         "text" tabindex="1">
         <BR>
         Имя:<INPUT name= "personal_firstname" type=
         "text" tabindex="2"> <BR>
      </FIELDSET>
      <FIELDSET>
      <LEGEND>Информация о почтовом ящике</LEGEND>
         Адрес:<INPUT name= "mail_address" tabindex="3'>
         <BR>
         Пароль:<INPUT name= "mail_password" type=
         "password" tabindex="4"> <BR>
         Подтверждение пароля <INPUT name=
         "mail_password" type= "password" tabindex="5">
      </FIELDSET>
      <FIELDSET>
      <LEGEND>Дополнительные сведения</LEGEND>
         Желаете получать рекламные ссылки?
         <INPUT name= "add_goods" type= "radio" value
         ="Yes" tabindex="6"> Да
```

```
      <INPUT name= "add_goods" type= "radio" checked
      value ="No" tabindex="7"> Нет <BR>
      Желаете получать прогноз погоды на каждый день?
      <INPUT name= "add_weather" type= "radio" value
      ="Yes" tabindex="8"> Да
      <INPUT name= "add_weather" type= "radio"
      checked value ="No" tabindex="9"> Нет
    </FIELDSET>
    <P>
      <INPUT type= "submit" value ="Регистрация"
      tabindex="10">
      <INPUT type= "reset" value ="Сброс" tabin-
      dex="11">
    </FORM>
  </BODY>
</HTML>
```

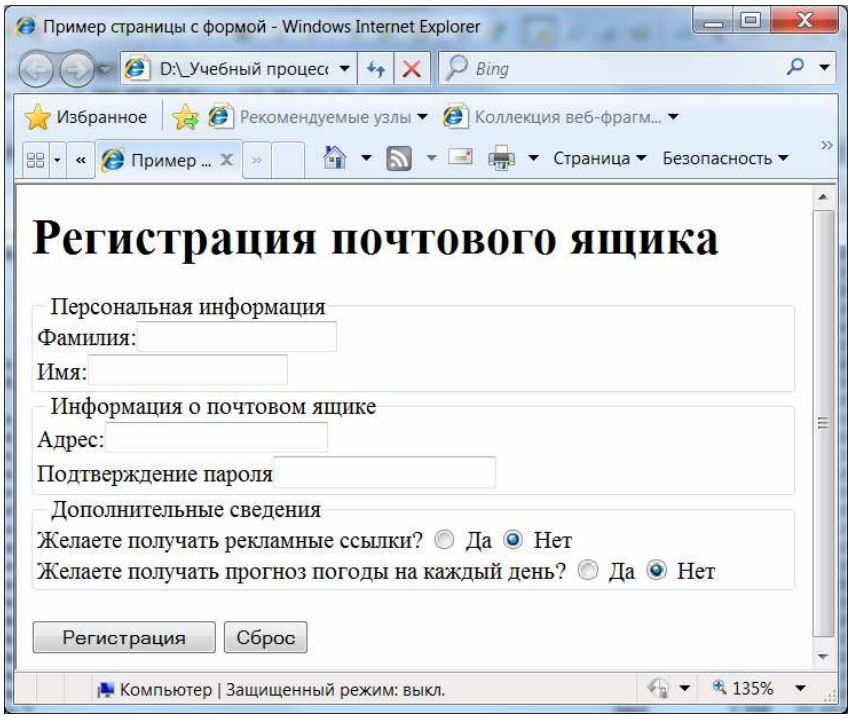


Рис. Л11.6. Результат выполнения задания № 6

Примечание. По адресу somesite.com/cgi-bin/proc.exe нет реального CGI-приложения, способного обрабатывать форму. Это просто пример практически законченной страницы с формой.

ЛАБОРАТОРНАЯ РАБОТА № 12
Гиперссылки в HTML-документах

Цель выполнения работы: изучение основных приемов создания гиперссылок в HTML-документах.

Теоретические сведения

Часть текста, название, участок изображения, имеющие ссылки на другой текст внутри этого документа или на другой документ в сети Интернет, называются гипертекстовой связью или гипертекстовой ссылкой.

Чтобы обычный документ превратить в гипертекстовый, в нем нужно разместить гипертекстовые связи. Для создания гиперссылки в тексте HTML-документа используется элемент A, который задается при помощи парных тегов <A> и </A>.

```
<A href=" "> … </A>
```

Атрибут href определяет URL-адрес ссылки, на которую будет совершен переход, если щелкнуть по объекту между тегами. Если ссылка производится на документ, находящийся в той же папке что и исходный документ, достаточно указать только имя файла.

Текст, изображение, или другой элемент HTML-документа, заключенный между этими тегами, становится представлением гиперссылки в тексте. Обычно браузеры отображают гиперссылки так, что их можно однозначно отделить от прочего содержимого документа, например, выделить цветом, подчеркиванием и т.д.

Обычно браузер устанавливает цветовое оформление ссылок по умолчанию, но можно изменить эти настройки специальными атрибутами тега <BODY>:

– link – цвет неактивированных ссылок;

– vlink – цвет посещенных ссылок;

– alink – цвет активной ссылки;

```
<BODY link = "green" alink = "lime" vlink = "gray">
```

Действием браузера при переходе по гиперссылке является открытие указанного HTML- документа в том же или новом окне.

Допустим, что сайт, по которому нужно реализовать навигацию, состоит из трех страниц. Они именуются 1.htm, 2.htm, 3.htm. У сайта также есть главная страница, на которой помещено оглавление. Элементы оглавления позволяют перейти к соответствующим страницам, то есть являются гиперссылками.

Тогда возможный вариант с оглавлением сайта будет выглядеть как в приведенном ниже задании.

Задание. Создание гиперссылок в HTML-документе

Создать три страницы по образцу, сохраняя под именами, соответственно: 1.htm, 2.htm, 3.htm.

1) Набрать в *Блокноте* приведенный ниже текст файла 1.htm. Открыть файл в браузере Internet Explorer и просмотреть результат (рис. Л12.1).

```
<HTML>
   <HEAD>
      <TITLE> Проектируемая страница 1</TITLE>
   </HEAD>
   <BODY >
      <H1><B>Содержимое первой страницы</B></H1>
      <A href="men.htm">Вернуться назад</A><BR>
</BODY>
</HTML>
```

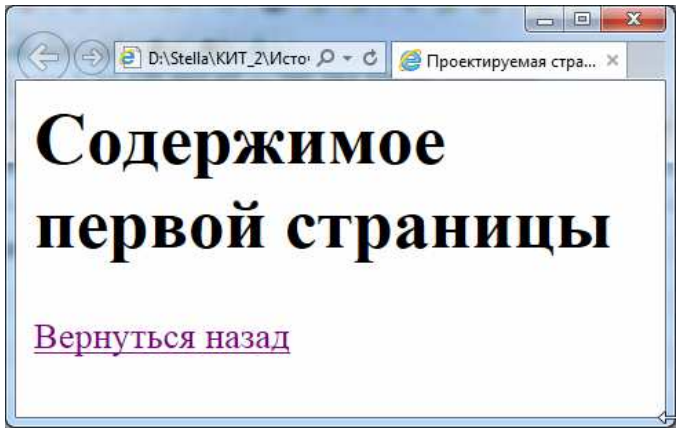


Рис Л12.1. Результат просмотра файла 1.htm

2) Набрать в *Блокноте* приведенный ниже текст файла 2.htm. Открыть файл в браузере Internet Explorer и просмотреть результат (рис. Л12.2).

```
<HTML>
   <HEAD>
      <TITLE> Проектируемая страница </TITLE>
   </HEAD>
   <BODY >
      <H2><B>Содержимое второй страницы</B></H2>
      <A href="men.htm">Вернуться назад</A><BR>
```

```
   </BODY>
</HTML>
```

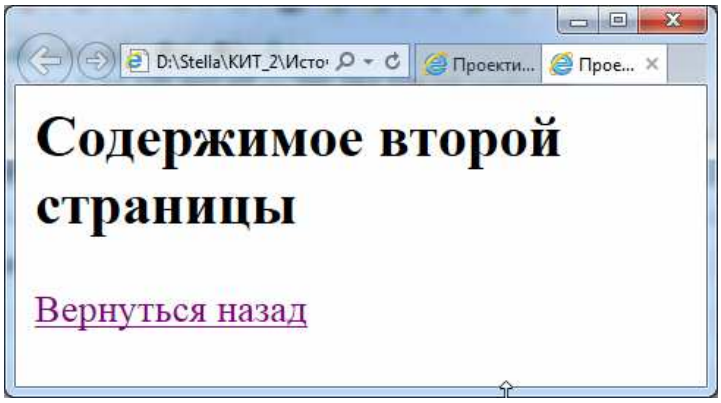


Рис Л12.2. Результат просмотра файла 2.htm

3) Набрать в *Блокноте* приведенный ниже текст файла 3.htm. Открыть файл в браузере Internet Explorer и просмотреть результат (рис. Л12.3).

```
<HTML>
   <HEAD>
      <TITLE> Проектируемая страница 3</TITLE>
   </HEAD>
   <BODY >
      <H3><B>Содержимое третьей страницы</B></H3>
      <A href="men.htm">Вернуться назад</A><BR>
   </BODY>
</HTML>
```

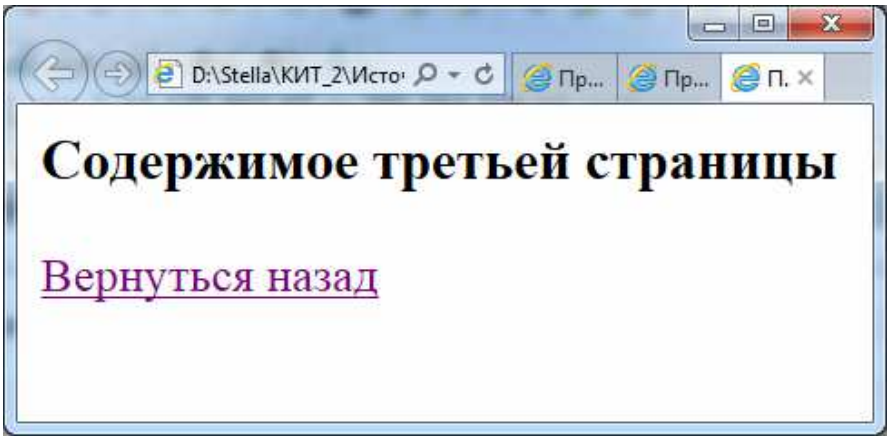


Рис Л12.3. Результат просмотра файла 3.htm

4) Набрать в *Блокноте* приведенный ниже текст файла men.htm. Открыть файл в браузере Internet Explorer и просмотреть результат (рис. Л12.4).

```
<HTML>
   <HEAD>
      <TITLE> Главная страница сайта</TITLE>
   </HEAD>
   <BODY>
   <DL>
   <DT><STRONG>ОГЛАВЛЕНИЕ</STRONG >
   <DD>
      <A href="1.htm">Первая страница</A><BR>
      <A href="2.htm">Вторая страница</A><BR>
      <A href="3.htm">Третья страница</A><BR>
   </DL>
   </BODY>
</HTML>
```

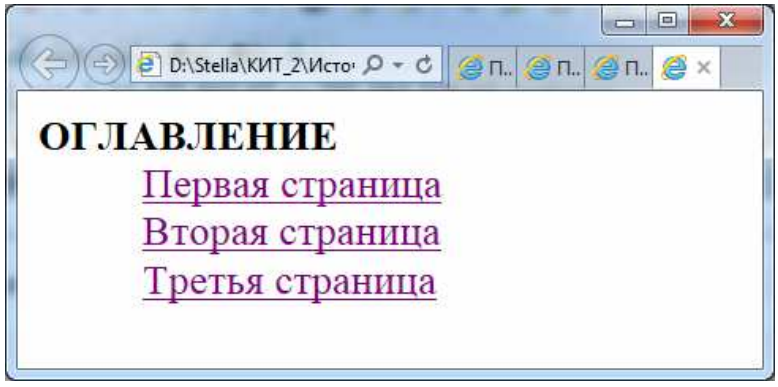


Рис Л12.4. Результат просмотра файла men.htm

5) Проверить, как функционируют созданные гиперссылки.

ЛАБОРАТОРНАЯ РАБОТА № 13
Фреймы в HTML-документах

Цель выполнения работы: изучение основных приемов создания фреймов в HTML-документах.

Теоретические сведения

Эта работа посвящена рассмотрению одного из самых интересных и удобных способов организации на веб-странице: использование фреймов.

Прежде рассматривалось создание отдельных HTML-документов. Примеры этой работы – создание сайтов, состоящих из нескольких страниц, так как использование фреймов предполагает наличие нескольких страниц, одновременно показываемых пользователю.

Фреймы – мощный механизм представления информации на веб-страницах. В один фрейм загружается список гиперссылок. При щелчке по любой из них содержимое новой страницы отображается в другом фрейме.

Для создания фреймовой структуры тег <BODY> *не используют*!

Вместо него необходим тег <FRAMESET> и парный ему тег </FRAMESET>.

Этот тег имеет следующие атрибуты:

– cols – количество фреймов-столбцов. Значением этого атрибута является список размеров каждого из фреймов;

Пример. Тег <FRAMESET COLS="20%, 60%, 20%”> задает три вертикальных фрейма соответствующих размеров. Размер последнего фрейма можно задавать с помощью символа *, обозначающего оставшуюся часть экрана. Размеры фреймов можно задавать не только в процентах, но и пикселях.

– rows – количество фреймов-строк;

Пример. Тег <FRAMESET rows="10%, *"> задает два горизонтальных фрейма, причем первый занимает 10% экрана, второй – всю оставшуюся часть. Если указать эти два атрибута одновременно, каждый горизонтальный фрейм будет разбит на указанное число вертикальных фреймов.

– border – ширина рамки между фреймами. При border="0" границы будут отсутствовать;

– bordercolor – цвет рамки.

Для описания каждого из фреймов используется тег <FRAME> с атрибутом src= "имя файла".

– `scrolling` – определяет наличие линеек прокрутки содержимого фрейма. Его возможные значения:

- `yes` – линейки прокрутки отображаются всегда;
- `no` – не отображаются;
- `auto` – появляются в случае необходимости.

Задание № 1. Создание фреймовой структуры с двумя вертикальными фреймами в HTML-документе

1) Набрать в *Блокноте* приведенный ниже текст файла left.htm. Открыть файл в браузере Internet Explorer и просмотреть результат (рис. Л13.1).

```
<HTML>
   <HEAD>
      <TITLE> Страница Левая</TITLE>
   </HEAD>
   <BODY>
      <H1 align=”center”> Левый</H1>
   </BODY>
</HTML>
```

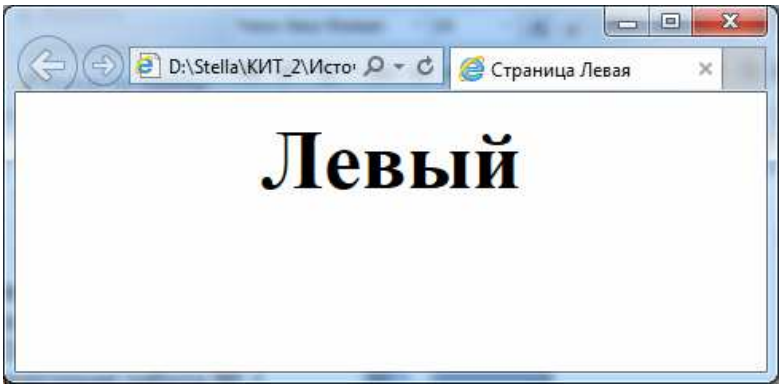


Рис. Л13.1. Результат просмотра файла left.htm

2) Набрать в *Блокноте* приведенный ниже текст файла right.htm. Открыть файл в браузере Internet Explorer и просмотреть результат (рис. Л13.2).

```
<HTML>
   <HEAD>
      <TITLE> Страница Правая</TITLE>
   </HEAD>
```

```
   <BODY>
      <H1 align=”center”> Правый</H1>
   </BODY>
</HTML>
```

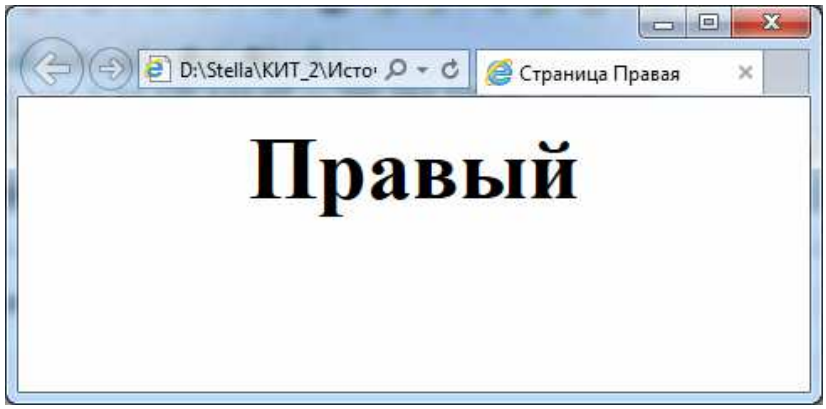


Рис. Л13.2. Результат просмотра файла right.htm

3) Набрать в *Блокноте* приведенный ниже текст файла F1.htm. Открыть файл в браузере Internet Explorer и просмотреть результат – фреймовую структуру с двумя вертикальными фреймами (рис. Л13.3).

```
<HTML>
        <HEAD>
        </HEAD>
        <FRAMESET COLS="50%,*">
           <FRAME SRC=left.htm>
           <FRAME SRC=right.htm>
        </FRAMESET>
</HTML>
```

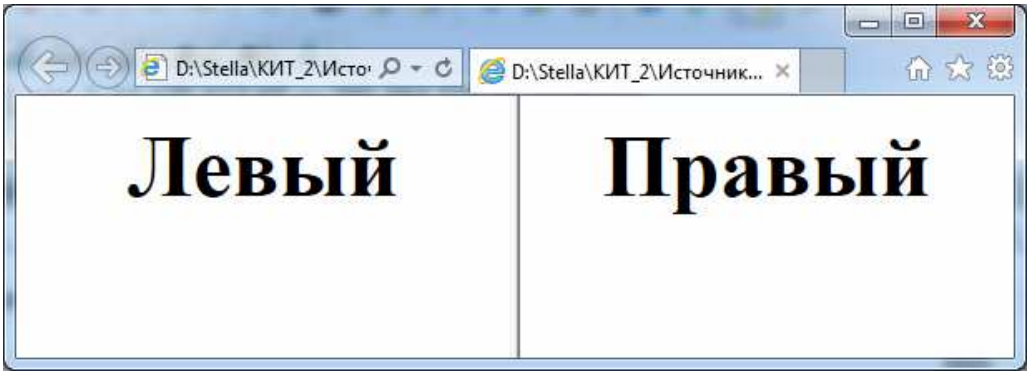


Рис. Л13.3. Результат просмотра файла F1.htm

Задание № 2. Создание фреймовой структуры с двумя горизонтальными фреймами в HTML-документе

1) Набрать в *Блокноте* приведенный ниже текст файла top.htm. Открыть файл в браузере Internet Explorer и просмотреть результат (рис. Л13.4).

```
<HTML>
   <HEAD>
      <TITLE> Страница Верх</TITLE>
   </HEAD>
   <BODY>
      <H1 align=”center”> Верх</H1>
   </BODY>
</HTML>
```

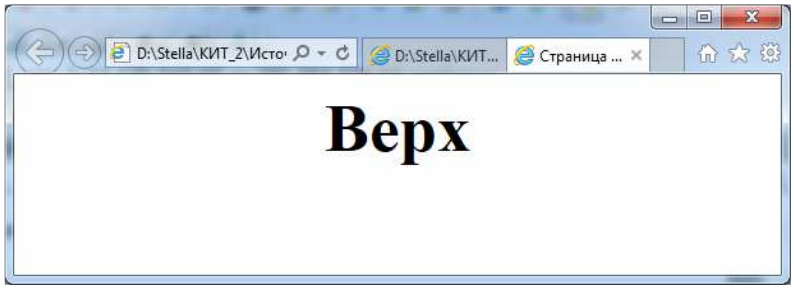


Рис. Л13.4. Результат просмотра файла top.htm

2) Набрать в *Блокноте* приведенный ниже текст файла bottom.htm. Открыть файл в браузере Internet Explorer и просмотреть результат (рис. Л13.5).

```
<HTML>
   <HEAD>
      <TITLE> Страница Низ</TITLE>
   </HEAD>
   <BODY>
      <H1 align=”center”> Низ</H1>
   </BODY>
</HTML>
```

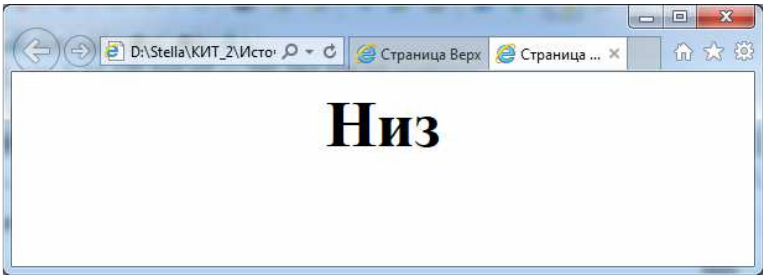


Рис. Л13.5. Результат просмотра файла bottom.htm

3) Набрать в *Блокноте* приведенный ниже текст файла F2.htm. Открыть файл в браузере Internet Explorer и просмотреть результат – фреймовую структуру с двумя горизонтальными фреймами (рис. Л13.6).

```
<HTML>
   <HEAD>
   </HEAD>
   <FRAMESET ROWS="50%,*">
      <FRAME SRC=top.htm>
      <FRAME SRC=bottom.htm>
   </FRAMESET>
</HTML>
```

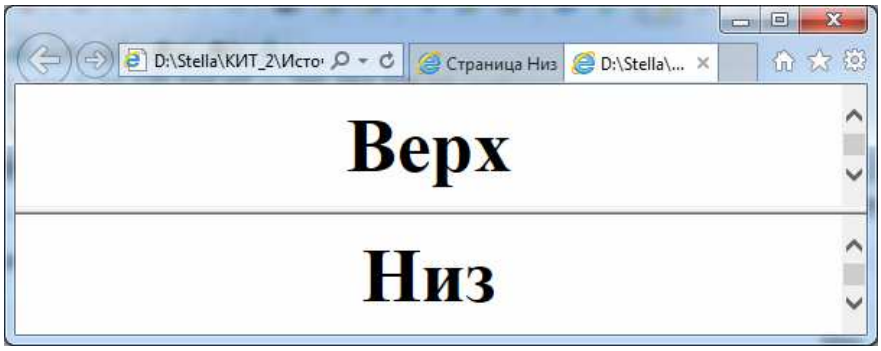


Рис. Л13.6. Результат просмотра файла F2.htm

Задание №3. Создание более сложных фреймовых структур в HTML-документе

1) Набрать в *Блокноте* приведенный ниже текст файла F3.htm. Открыть файл в браузере Internet Explorer и просмотреть результат – фреймовую структуру с разделенным по горизонтали правым фреймом (рис. Л13.7).

```
<HTML>
   <HEAD>
   </HEAD>
   <FRAMESET COLS="50%,*">
      <FRAME NAME=left SRC=left.htm>
   <FRAMESET ROWS="50%,*">
      <FRAME SRC=top.htm>
      <FRAME SRC=bottom.htm>
   </FRAMESET>
   </FRAMESET>
</HTML>
```

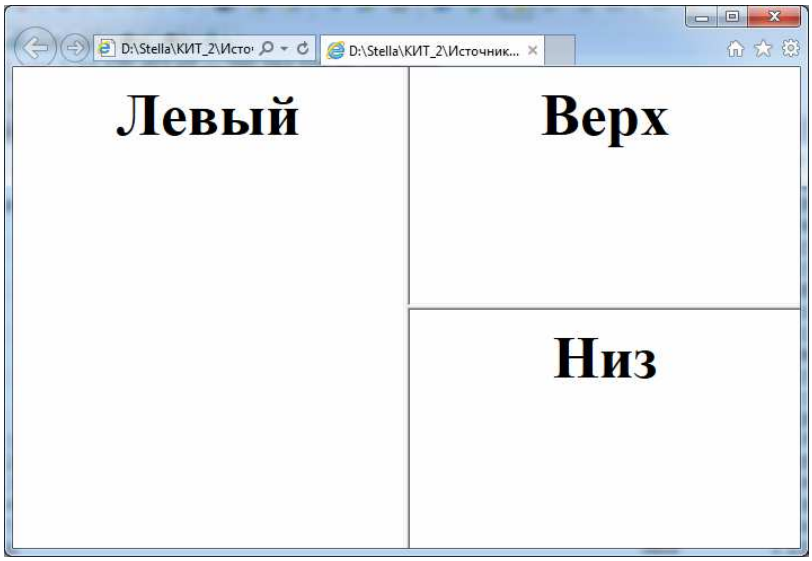


Рис. Л13.7. Результат просмотра файла F3.htm

2) Набрать в *Блокноте* приведенный ниже текст файла centr.htm. Открыть файл в браузере Internet Explorer и просмотреть результат (рис. Л13.8).

```
<HTML>
   <HEAD>
   <TITLE> Страница центр </TITLE>
   </HEAD>
   <BODY>
      <H1> Центр</H1>
   </BODY>
</HTML>
```

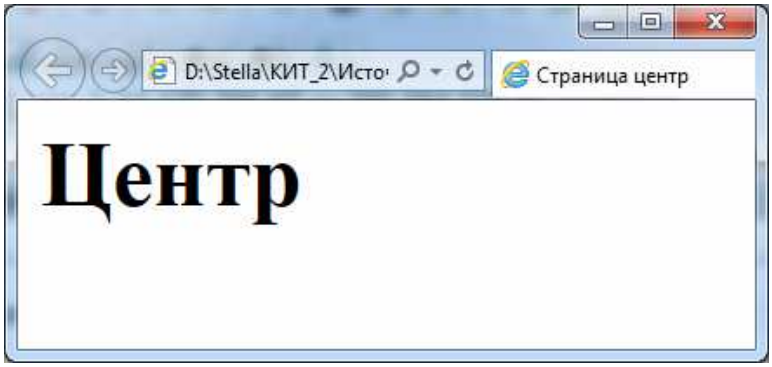


Рис. Л13.8. Результат просмотра файла centr.htm

3) Набрать в *Блокноте* приведенный ниже текст файла F4.htm. Открыть файл в браузере Internet Explorer и просмотреть результат – фреймовую структуру с разделенным по вертикали на три части центральным фреймом (рис. Л13.9).

```
<HTML>
   <FRAMESET ROWS="50,*,50">
      <FRAME SRC=top.htm>
   FRAMESET COLS="100,*,100">
      <FRAME SRC=left.htm>
      <FRAME SRC=centr.htm>
      <FRAME SRC=right.htm>
   </FRAMESET>
      <FRAME SRC=bottom.htm>
   </FRAMESET>
</HTML>
```

Рис. Л13.9. Результат просмотра файла F4.htm

ВОПРОСЫ И ЗАДАНИЯ К ЭКЗАМЕНУ

Основы программирования на Visual Basic for Applications в Microsoft Excel

1. Что такое язык программирования?
2. Что такое программа?
3. Что представляет собой процесс программирования?
4. Из каких стадий состоит процесс программирования?
5. Охарактеризуйте различные уровни языков программирования.
6. Охарактеризуйте группы языков программирования.
7. Охарактеризуйте основные свойства объекто-ориентированного программирования (ООП): наследование, инкапсуляцию, полиморфизм.
8. Каковы преимущества ООП при создании больших программ?
9. Что такое Visual Basic for Applications?
10. Каковы принципы разработки приложений электронных таблиц?
11. Какие средства создания пользовательского интерфейса существуют в Microsoft Excel? Охарактеризуйте их.
12. Как ведется работа с конечным пользователем при создании приложения электронных таблиц?
13. Что представляют собой объекты Microsoft Excel? Охарактеризуйте свойства, методы и события объектов в Excel VBA.
14. Перечислите и охарактеризуйте компоненты редактора Visual Basic.
15. Как ведется настройка среды VBE?
16. Как в редакторе Visual Basic выполняется ввод программного кода? Какие рекомендации существуют на этот счет?
17. Какие математические операторы используются в VBA?
18. С данными каких типов работает VBA? Как объявляются константы и переменные, и как с ними ведется в VBA работа?
19. Как в VBA выполняется обработка строковых данных и дат?
20. Что такое массив? Как в VBA объявляются одномерные и многомерные массивы?
21. Что представляют собой переменные объектов?
22. Для чего используются пользовательские типы данных?
23. Как в VBA ведется работа со встроенными функциями?
24. Как в VBA осуществляется управление объектами и коллекциями?
25. Как в VBA осуществляется контроль выполнения программного кода?
26. Что такое процедура? Как в VBA объявляются процедуры?
27. Как в VBA выполняется обработка ошибок?
28. Как в VBA создаются пользовательские формы (собственные диалоговые окна)?
29. Как в VBA ведется работа с пользовательскими диалоговыми окнами?

Сетевые информационные технологии

1. Назначение компьютерных сетей.
2. Классификация компьютерных сетей по территориальной распространенности.
3. Классификация компьютерных сетей по скорости передачи данных.
4. Классификация компьютерных сетей по топологии.
5. Классификация компьютерных сетей по способу управления.
6. Классификация компьютерных сетей по методу коммутации.
7. Стандартизация компьютерных сетей. Понятия протокола и стека протоколов.
8. Стек проколов TCP/IP.
9. Интернет, основные понятия.
10. История развития Интернета.
11. Адресация в Интернете: домены.
12. Адресация в Интернете: система адресации URL.
13. Службы Интернета: Telnet, электронная почта, Usenet, World Wide Web, служба имен доменов, служба передачи файлов, ICQ, списки рассылки.
14. Поиск информации в Интернете: индексированные каталоги.
15. Поиск информации в Интернете: поисковые машины.
16. Поиск информации в Интернете: онлайновые справочники и энциклопедии.
17. Что представляет собой язык гипертекстовой разметки HTML? Каково его назначение?
18. Перечислите и охарактеризуйте виды веб-страниц.
19. Какова обобщенная структура html-документа?
20. Теги HTML, их классификация. Синтаксис наиболее часто используемых тегов.
21. Каковы этапы создания Web-сайта?
22. Перечислите и охарактеризуйте основные программы для создания Web-документов.
23. Какие преимущества дает использование интернет-технологий в бизнесе?
24. Дайте характеристику информационных витрин и информационных порталов.
25. Что такое электронный бизнес?
26. Что такое электронная коммерция? Каковы ее достоинства и недостатки?
27. Охарактеризуйте сектор электронной коммерции B2B.
28. Охарактеризуйте сектор электронной коммерции B2C.
29. Охарактеризуйте сектор электронной коммерции C2B.
30. Охарактеризуйте сектор электронной коммерции C2C.
31. Что представляет собой мобильная коммерция?

НЕКОТОРЫЕ ПРИМЕРЫ ТЕСТОВЫХ ЭКЗАМЕНАЦИОННЫХ ЗАДАНИЙ

Тема «ОСНОВЫ ПРОГРАММИРОВАНИЯ НА VISUAL BASIC FOR APPLICATIONS В MICROSOFT EXCEL»

1. Язык программирования – это…
   – набор символов и терминов, который в соответствии с правилами синтаксиса описывает алгоритм решения задачи;
   – совокупность операторов, записанных в соответствии с принятым синтаксисом;
   – последовательность действий (операций), проводимая в целях достижения требуемого результата.
2. Какие уровни языков программирования существуют?
   – машинные;
   – не машинные;
   – машинно-ориентированные;
   – машинно-независимые.
3. Пользовательский интерфейс – это…
   – метод взаимодействия пользователя с приложением;
   – метод взаимодействия приложения с другими приложениями;
   – метод взаимодействия приложения с устройствами компьютера.
4. Что изображено на рисунке?

   – панель инструментов «Веб-узел»;
   – панель инструментов «Visual Basic»;
   – панель инструментов «Элементы управления»
5. Объект в Microsoft Excel – это…
   – любой элемент приложения;
   – любой рабочий лист;
   – выделенный диапазон ячеек;
   – ячейка, содержащая формулу.

6. Для обозначения строки комментария в VBA можно использовать следующие средства:
   – #;
   – ‘;
   – `Rem`;
   – `Sub`.
7. Для объявления переменных в VBA используется оператор…
   – `Dim`;
   – `Sub`;
   – `Rem`.

8. В VBA оператором присвоения выступает знак…
   – ~;
   – #;
   – ≈;
   – ≅;
   – =.
9. Массив – это…
   – группа элементов разных типов, которые имеют общее имя;
   – группа элементов разных типов, которые имеют различные имена;
   – группа элементов одного типа, которые имеют общее имя;
   – группа элементов одного типа, которые имеют различные имена.
10. Какие конструкции из приведенных ниже используются для программирования ветвлений?
   – `If-Then`;
   – `Select Case`;
   – `For-Next`;
   – `Do While`;
   – `Do Until`.
11. Какие конструкции из приведенных ниже используются для программирования циклов?
   – `If-Then`;
   – `Select Case`;
   – `For-Next`;
   – `Do While`;
   – `Do Until`.
12. Для создания окна ввода в VBA используется функция…
   – `MsgList`;
   – `MsgBox`;
   – `InputList`;
   – `InputBox`.

Тема «СЕТЕВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

1. Интернет – это…
   – локальная вычислительная сеть;
   – глобальная компьютерная сеть, состоящая из множества соединенных друг с другом меньших по размеру сетей и покрывающая весь земной шар;
   – региональная информационно-вычислительная сеть;
   – сетевая операционная система.

2. Служба Telnet предназначена для…
   – работы с IPX (Internet work Packet exchange – межсетевая передача пакетов);
   – работы с операционной системой Windows;
   – удаленного управления компьютером;
   – проведения телеконференций.
3. Назначение службы E-Mail…
   – просмотр страниц WWW;
   – удаленное управление компьютером;
   – электронная почта;
   – работы с IPX (Internet work Packet exchange – межсетевая передача пакетов).
4. Служба Usenet предназначена для…
   – проведения телеконференций;
   – просмотр страниц WWW;
   – удаленного управления компьютером;
   – связи между собой нескольких локальных сетей, работающих по разным протоколам.
5. World Wide Web – это…
   – служба удаленного управления компьютером;
   – единое информационное пространство, состоящее из взаимосвязанных электронных документов, хранящихся на Web-серверах;
   – служба телеконференций;
   – глобальная компьютерная сеть, состоящая из множества соединенных друг с другом меньших по размеру сетей и покрывающая весь земной шар.
6. Web-страницы – это…
   – текстовые документы;
   – документы в формате RTF;
   – наибольшие единицы Всемирной информационной сети;
   – отдельные документы, составляющие пространство Web.
7. Гиперссылка – это…
   – выделенный фрагмент документа, с которым ассоциирован адрес другого Web-документа;
   – текст, созданный на страницах WWW с помощью программы Netscape Navigator;
   – текст, созданный на страницах WWW с помощью программы Microsoft Internet Assistant for Word;
   – иллюстрация на странице WWW.
8. Перемещение между Web-документами называют…
   – Web-перемещением;
   – Web-путешествием;

– Web-навигацией;
– Web-круизом.

9. Программы для просмотра Web-страниц называют…
– брандмауэрами;
– браузерами;
– WWW-навигаторами;
– Интернет-навигаторами.

10. Каждому документу в Интернете присваиваются (указать все правильные ответы)…
– адрес URL;
– IP-адрес;
– WWW-адрес;
– http-адрес.

11. Служба DNS предназначена для …
– удаленного управления компьютером;
– просмотр страниц WWW;
– работы с IPX (Internet work Packet exchange – межсетевая передача пакетов);
– перевода доменных имен в связанные с ними IP-адреса.

12. Служба ICQ предназначена для…
– поиска сетевого *IP*-адреса человека, подключенного в данный момент к Интернету;
– работы с IPX (Internet work Packet exchange – межсетевая передача пакетов);
– перевода доменных имен в связанные с ними IP-адреса;
– просмотр страниц WWW.

13. Какое из перечисленных приложений не является браузером?
– Internet Explorer;
– Netscape Navigator;
– Opera;
– Adobe Acrobat.

14. Помощь в поиске нужной информации в Интернете оказывают (указать все правильные ответы)…
– браузеры;
– поисковые каталоги;
– поисковые предметные указатели;
– поисковые машины;
– мастера поиска;
– ключевые слова.

ЛИТЕРАТУРА

1. Волченков, Н. Г. Программирование на Visual Basic 6 : в 3-х ч. / Н. Г. Волченков. – М. : ИНФРА-М, 2000.

2. Гарнаев, А. Ю. Использование MS Excel и VBA в экономике и финансах / А. Ю. Гарнаев. – СПб. : БХВ-Петербург, 2000. – 336 с.

3. Гарнаев, А. Ю. Самоучитель VBA / А. Ю. Гарнаев. – СПб. : БХВ-Петербург, 2007. – 512 с.

4. Демидова, Л. А. Программирование в среде Visual Basic for Applications : практикум / Л. А. Демидова, А. Н. Пылькин. – М. : Горячая линия-Телеком, 2004. – 175 с.

5. Кашаев, С. М. Офисные решения с использованием Microsoft Excel 2007 и VBA / С. М. Кашаев. – СПб. : Питер, 2009.

6. Сафронов, И. К. Visual Basic в задачах и примерах / И. К. Сафронов. – СПб. : БХВ-Петербург, 2007. – 400 с.

7. Слепцова, Л. Д. Программирование на VBA в Microsoft Office 2007 : самоучитель / Л. Д. Слепцова. – М. : Вильямс, 2007. – 432 с.

8. Уокенбах, Д. Профессиональное программирование на VBA в Excel 2002 : пер. с англ. / Д. Уокенбах. – М. : Вильямс, 2003. – 784 с.

9. Кузнецов, С. Интернет в бизнесе и бизнес в Интернете [Электронный ресурс] / С. Кузнецов. – Режим доступа: www.citforum.ru/internet/.

Учебное издание

КОМПЬЮТЕРНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Учебно-методический комплекс
для студентов экономических специальностей

В трех частях

Часть 2

РЯСОВА Стелла Евгеньевна
ОСЬКИН Дмитрий Аркадьевич

**Основы программирования
на Visual Basic For Applications в Microsoft Excel.
Сетевые инфрмационные технологии**

Редактор *Д. М. Севастьянова*
Дизайн обложки *Е. Н. Бурцевой*

Подписано в печать 02.05.2016. Формат 60×84 1/16. Бумага офсетная.
Ризография. Усл. печ. л. 18,56. Уч.-изд. л. 16,92. Тираж 30 экз. Заказ 857.

Издатель и полиграфическое исполнение:
учреждение образования «Полоцкий государственный университет».
Свидетельство о государственной регистрации
издателя, изготовителя, распространителя печатных изданий
№ 1/305 от 22.04.2014.

ЛП № 02330/278 от 08.05.2014.

Ул. Блохина, 29, 211440, г. Новополоцк.