

Министерство образования Республики Беларусь
ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра технологий программирования

**Методические указания
к лабораторной работе № 8
по курсу «Основы алгоритмизации
и программирования»**

«Строки»

Преподаватель: Войтехович
Агния Витольдовна

Полоцк, 2015

ЦЕЛЬ РАБОТЫ

Научиться работать с векторными данными языка «С»: строки. Изучить правила объявления строки и обращение к их элементам, библиотечные функции для работы со строками. Освоить основные алгоритмы обработки строк.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1. Формат строки, объявление и инициализация

Исторически сложилось два представления формата строк:

- формат ANSI;
- строки с завершающим нулем (используется в языке С).

Формат ANSI устанавливает, что значением первой позиции в строке является ее длина, а затем следуют сами символы строки. Например, представление строки “Моя строка!” будет следующим:

| | | | | | | | | | | | |
|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 11 | 'M' | 'o' | 'я' | ' ' | 'c' | 'т' | 'р' | 'о' | 'к' | 'а' | '!' |
|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|

В строках с завершающим нулем, значащие символы строки указываются с первого позиции, а признаком завершения строки является значение ноль. Представление рассмотренной ранее строки в этом формате имеет вид:

| | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|---|
| 'M' | 'o' | 'я' | ' ' | 'c' | 'т' | 'р' | 'о' | 'к' | 'а' | '!' | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|---|

Как уже отмечалось, строки в языке С реализуются посредством массивов символов. Поэтому объявление ASCII строки на языке С имеет следующий синтаксис:

char имя [длина];

Таким образом, объявление строки имеет тот же синтаксис, что и объявление одномерного символьного массива. Длина строки должна представлять собой целочисленное значение (в стандарте **C89** - константа, в стандарте **C99** может быть выражением). Длина строки указывается с учетом одного символа на хранение завершающего нуля, поэтому максимальное количество значащих символов в строке на единицу меньше ее длины. Например, строка может содержать максимально двадцать символов, если объявлена следующим образом:

char str[21];

Инициализация строки осуществляется при ее объявлении, используя следующий синтаксис:

char str[длина] = строковый литерал;

Строковый литерал - строка ASCII символов заключенных в двойные кавычки.

Примеры объявления строк с инициализацией:

```
char str1[20] = "Введите значение: ",  
      str2[20] = "";
```

Строка *str1* объявлена с инициализирующей строкой “Введите значение: ”, а строка *str2* - с пустой строкой. Длина инициализирующей строки должна быть строго меньше длины объявляемой строки (для учета символа завершающего нуля).

Объявление константных строковых переменных начинается с ключевого слова **const**, за которым следует объявление строки с инициализацией. Пример:

```
const char message[] = "Сообщение об ошибке!";
```

Так как строки на языке С являются массивами символов, то к любому символу строки можно обратиться по его индексу. Для этого используется синтаксис обращения к элементу массива, поэтому первый символ в строке имеет индекс ноль. Например, в следующем фрагменте программы в строке *str* осуществляется замена всех символов ‘*a*’ на символы ‘*A*’ и наоборот.

```
for(int i=0; str[i]!=0; i++) {  
    if(str[i] == 'a') str[i] = 'A';  
    else if(str[i] == 'A') str[i] = 'a';  
}
```

В программе используется цикл **for** с переменной-счетчиком *i*. Цикл выполняется со значения первого символа строки. Выход из цикла осуществляется при достижении конца строки (*i*-ый символ строки равен нулю). В теле цикла в первом операторе **if** выполняется сравнение значения текущего символа строки с символом ‘*a*’, и, если равен, то осуществляется его замена на символ ‘*A*’. В противном случае во втором операторе **if** осуществляется сравнение с символом ‘*A*’, и, при положительном исходе, осуществляется замена на символ ‘*a*’.

Объявление массивов строк в языке С также возможно. Для этого используются двумерные массивы символов, что имеет следующий синтаксис:

```
char имя [количество] [длина];
```

Первым размером матрицы указывается количество строк в массиве, а вторым - максимальная (с учетом завершающего нуля) длина каждой строки. Например, объявление массива из пяти строк максимальной длиной 30 значащих символов будет иметь вид:

```
char strs[5][31];
```

При объявлении массивов строк можно производить инициализацию:

```
char имя [количество] [длина] =  
    {строковый литерал №1, ... строковый литерал №N};
```

Число строковых литералов должно быть меньше или равно количеству строк в массиве. Если число строковых литералов меньше размера массива, то все остальные элементы инициализируются пустыми строками. Длина каждого строкового литерала должна быть строго меньше значения длины строки (для записи завершающего нуля). Например:

```
char days[12][10] =  
{"Январь", "Февраль", "Март", "Апрель", "Май", "Июнь",  
"Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь",  
"Декабрь"};
```

Соответственно, при объявлении массивов строк с инициализацией допускается не указывать количество строк в квадратных скобках. В таком случае, количество строк в массиве будет определено автоматически по числу инициализирующих строковых литералов. Например, массив из семи строк:

```
char days[] [12] =  
{"Понедельник", "Вторник", "Среда", "Четверг",  
"Пятница", "Суббота", "Воскресенье"};
```

Никаких других операторов для работы со строками в языке С не предусмотрено. Вся обработка строк осуществляется посредством различных функций библиотек языка С.

2. Функции для работы со строками

Все библиотечные функции, предназначенные для работы со строками, можно разделить на три группы:

- ввод и вывод строк;
- преобразование строк;
- обработка строк.

Для ввода и вывода строковой информации можно использовать функции форматированного ввода и вывода (**printf** и **scanf**). Для этого в строке формата при вводе или выводе строковой переменной необходимо указать спецификатор типа **%s**. Например, ввод и последующий вывод строковой переменной будет иметь вид:

```
char str[31] = "";  
printf("Введите строку: ");  
scanf("%s", str);  
printf("Вы ввели: %s", str);
```

В первой строке осуществляется объявление строковой переменной *str* максимальной длиной 30 символов и инициализацией пустой строкой. Далее выводится приглашение к вводу строки и, затем, осуществляется непосредственно ввод. Следует обратить внимание на то, что в функции **scanf** имя переменной *str* указано без знака амперсанда. Такое допустимо только при вводе строк, так как

имя строковой переменной является именем массива, а имя массива в языке С является указателем на сам массив (указатели будут рассматриваться в следующее главе). Следовательно, имя строки, по сути, является указателем на эту строку и знак амперсанда можно не указывать. В последней строке осуществляется вывод введенного значения.

Недостатком функции **scanf** при вводе строковых данных является то, что символами разделителями данной функции являются: перевод строки, табуляция и пробел. Поэтому, используя данную функцию невозможно ввести строку, содержащую несколько слов, разделенных пробелами или табуляциями. Например, если в предыдущей программе пользователь введет строку "Сообщение из нескольких слов", то на экране будет выведено только "Сообщение". Поэтому для ввода и вывода строк в библиотеке **stdio.h** содержатся специализированные функции **gets** и **puts**.

ПРИМЕЧАНИЕ: При работе со строками в языке С активно используются указатели. Если возникает необходимость, рекомендуется сначала обратиться к следующей главе.

Функция **gets** предназначена для ввода строк и имеет следующий заголовок:

```
char * gets(char *buffer);
```

Единственным параметром функции является строковая переменная, в которую будут вводиться данные. Функция возвращает значение параметра *buffer*, если строка была введена, или значение **NULL** (ноль), если был достигнут конец потока ввода. Данная функция осуществляет ввод строки, пока не встретится символ перевода строки.

ПРИМЕЧАНИЕ: Ввод и вывод в языке С осуществляется посредством потоков. Понятие потоков и все, что с ними связано, будет рассматриваться в главе 7.

Недостатком функции **gets** является то, что невозможно контролировать длину вводимой строки, вследствие чего возможно переполнение отведенного под строку пространства памяти, что по своей сути является критической ошибкой. Для предотвращения подобной ситуации следует использовать функцию **fgets** (будет рассматриваться в главе 7).

Функция **puts** предназначена для вывода строк и имеет следующий заголовок:

```
int puts(const char *string);
```

Единственным параметром функции является строковая переменная, значение которой необходимо вывести. Функция возвращает целочисленное значение: положительное число или ноль в случае успеха, и значение **EOF** (-1) в случае ошибки. Данная функция автоматически выводит в поток символ перевода строки.

Простейшая программа: ввод и вывод строки с использованием функций **gets** и **puts** будет иметь вид:

```
char str[100] = "";
printf("Введите строку: "); gets(str);
printf("Вы ввели: "); puts(str);
```

Помимо функций ввода и вывода в потоки в библиотеке **stdio.h** присутствуют функции форматированного ввода и вывода в строки. Функция форматированного ввода из строки имеет следующий заголовок:

```
int sscanf(const char * restrict buffer,
           const char * restrict string, [address] ...);
```

Первым параметром функции является строка *buffer*, из которой будет производиться считывание данных согласно формату ввода. Остальные параметры и возвращаемое значение функции аналогичны параметрам и возвращаемому значению функции **scanf**.

Функции форматированного вывода в строку имеют следующие заголовки:

```
int sprintf(char * restrict buffer,
            const char * restrict format, [argument] ...);
int snprintf(char * restrict buffer, size_t maxlen,
             const char * restrict format, [argument] ...);
```

В параметре *buffer* передается строка, в которую будет осуществляться вывод. В функции **snprintf** вторым параметром *maxlen* указывается максимальное число символов, которое может быть записано в результирующую строку (остальные символы просто отбрасываются). Параметры *format* и *argument* аналогичны одноименным параметрам функции **printf**. Возвращаемое значение и поведение функций также аналогично функции **printf**. Функция **snprintf** была введена только в стандарте **C99**.

Рассмотренные функции **sscanf**, **sprintf** и **snprintf** очень удобны для преобразования численных данных в строковые и наоборот. Например, в следующем фрагменте программы осуществляется ввод целых чисел и вычисление их суммы (ввод значений продолжается пока не будет введена пустая строка):

```
int summa = 0;
while(1) {
    char str[15];
    printf("Введите число или пустую строку: ");
    gets(str);
    if(str[0]==0) break;
    int n;
    sscanf(str,"%d", &n);
    summa += n;
}
printf("Сумма чисел: %d\n", summa);
```

В первой строке осуществляется объявление и инициализация целочисленной переменной *summa*, в которой будет накапливаться сумма вводимых чисел. Далее описывается бесконечный цикл **while**. В теле цикла объявляется строка *str*, выводится приглашение к вводу данных и осуществляется ввод строки. Затем в операторе **if** осуществляется проверка, является ли введенное значение пустой строкой. Если да, то осуществляется выход из цикла посредством оператора **break**. Проверка осуществляется исходя из того, что первый символ пустой строки есть завершающий ноль. Если введена не пустая строка, то объявляется переменная *n*, в которую посредством функции *sscanf* записывается введенное значение (фактически здесь осуществляется преобразование строки в число) и вычисляется сумма. После цикла осуществляется вывод суммы.

В предыдущем примере были рассмотрено использование функции форматированного ввода из строки для преобразования строки в целое число. Тем не менее, данный пример является небезопасным: если введено не целое число, а какая-нибудь строка, то результат программы не предсказуем. Поэтому для преобразования строк, содержащих числа, в численные значения в библиотеке **stdlib.h** предусмотрен следующий набор функций:

```
double atof(const char *string);
int atoi(const char *string);
long int atol(const char *string);
long long int atoll(const char *string);
```

В качестве параметра во всех функциях выступает строка *string*, которая содержит текстовое представление числа в десятичной системе исчисления. Функции возвращают результат в виде числа определенного типа (вещественное - **atof**, целое - **atoi**, длинное целое - **atol**, длинное длинное целое - **atoll**). В случае переполнения результат выполнения функций не определен. В случае некорректного представления числа функции возвращают нулевые значения соответствующего формата. Корректное представление вещественного числа в текстовой строке должно удовлетворять формату:

[пробел] [+|-] [цифры] [.цифры] [{d|D|e|E} [+|-]} цифры]

После символов D, d, E, e указывается порядок числа. Корректное представление целого числа в текстовой строке должно удовлетворять формату:

[пробел] [+|-] цифры

Функции осуществляют преобразование строки, пока не встретится символ неудовлетворяющий формату, или пока не будет достигнут конец строки.

Помимо приведенных выше функций в библиотеке **stdlib.h** доступны также следующие функции преобразования строк в вещественные числа:

```
float strtod(const char * restrict string,
             char ** restrict endptr);
```

```
double strtod(const char * restrict string,
              char ** restrict endptr);
long double strtold(const char * restrict string,
                     char ** restrict endptr);
```

Приведенные функции осуществляют преобразование строк, содержащих вещественные числа, в вещественные значения типа **float**, **double** и **long double** соответственно. В первом параметре функций передается строка *string*, содержащая текстовое представление вещественного числа. Если второй параметр *endptr* не **NULL**, то через него осуществляется возврат указателя на не преобразованное окончание строки. Функции возвращают значение ноль, если строка не может быть интерпретирована как вещественное число. При переполнении функции возвращают значения **+/-HUGE_VALF / HUGE_VAL / HUGE_VALL** соответственно.

ПРИМЕЧАНИЕ: Для передачи второго параметра и возврата через него некоторого значения используется принцип передачи параметров по ссылке. Описание данного принципа будет дано в главе 6.

Аналогичные функции присутствуют и для преобразования строк в целочисленные значения:

```
long int strtol(const char * restrict string,
                char ** restrict endptr, int base);
unsigned long strtoul(const char * restrict string,
                      char ** restrict endptr, int base);
long long int strtoll(const char * restrict string,
                      char ** restrict endptr, int base);
unsigned long long strtoull(
    const char * restrict string,
    char ** restrict endptr, int base);
```

Приведенные функции осуществляют преобразование строк, содержащих целые числа, в целочисленные значения типа **long int**, **unsigned long**, **long long int** и **unsigned long long int** соответственно. В первом параметре функций передается строка *string*, содержащая текстовое представление целого числа. Если второй параметр *endptr* не **NULL**, то через него осуществляется возврат указателя на не преобразованное окончание строки. В третьем параметре *base* передается основание системы исчисления, в котором записано число. Функции возвращают значение ноль, если строка не может быть интерпретирована как целое число. При переполнении функции возвращают значения **+/-LONG_MAX / LLONG_MAX** или **ULONG_MAX / ULLONG_MAX** соответственно.

Функции обратного преобразования (численные значения в строки) в библиотеке **stdlib.h** присутствуют, но они не регламентированы стандартом, и рассматриваться не будут. Для преобразования численных значений в строковые наиболее удобно использовать функции **sprintf** и **snprintf**.

В библиотеке **string.h** содержаться функции для различных действий над строками. В данном руководстве будут рассмотрены только основные из них, входящие в стандарт языка.

Функция вычисления длины строки:

```
size_t strlen(const char *string);
```

В качестве параметра передается строка *string*. Функция возвращает численное значение длины строки (тип **unsigned int**). Пример:

```
char str[] = "1234";
int n = strlen(str); //n == 4
```

Функции копирования строк:

```
char * strcpy(char * restrict dst,
              const char * restrict src);
char * strncpy(char * restrict dst,
               const char * restrict src, size_t num);
```

Первым параметром *dst* является строковая переменная, в которую необходимо скопировать строковое значение (назначение), а вторым параметром - строковая переменная значение которой будет копироваться (источник). Третий параметр *num* функции **strncpy** указывает, какое число символов от начала строки необходимо скопировать. Если длина строки источника больше значения параметра *num*, то в строку назначение копируется только первых *num* символов строки источника, в противном случае - вся строка. Функции возвращают фактическое значение параметра *dst*. Пример:

```
char str[] = "abcdefg", dst1[10] = "", dst2[10] = "";
strcpy(dst1,str);           //dst1 == "abcdefg"
strncpy(dst2,str,5);       //dst2 == "abcde"
```

Функции сравнения строк:

```
int strcmp(const char *string1, const char *string2);
int strncmp(const char *string1, const char *string2,
            size_t num);
```

Первым *string1* и вторым *string2* параметрами функции являются строки, значения которых необходимо сравнить. Третьим параметром *num* функции **strncmp** указывается, какое число символов от начала строк необходимо сравнивать. Функции осуществляют сравнение строк по алфавитному порядку и возвращают:

- положительное значение - если *string1* больше *string2*;
- отрицательное значение - если *string1* меньше *string2*;
- нулевое значение - если *string1* совпадает с *string2*.

Пример:

```
char str1[] = "Компьютер", str2[] = "Компьютерный";
int n1 = strcmp(str1,str2);           //n1 < 0
int n2 = strncmp(str1,str2,9);       //n2 == 0
```

Функции объединения (конкатенации) строк:

```
char * strcat(char * restrict dst,
              const char * restrict src);
char * strncat(char * restrict dst,
               const char * restrict src, size_t num);
```

Первыми двумя параметрами функций являются строки: *dst* - назначение, *src* - источник. В третьем параметре *num* функции **strncat** указывается, какое число первых символов строки *src* следует добавить к строке *dst*. Функции объединения строк не создают новую строковую переменную, поэтому при написании программ необходимо проверять возможное переполнение строки *dst*. Если длина строки источника больше значения параметра *num*, то к строке назначение прибавляется только первых *num* символов строки источника, в противном случае - вся строка. Функции возвращают фактическое значение параметра *dst*. Пример:

```
char str1[20] = "серверный ", str2[20] = "Главный ",
      str3[] = "зал";
strncat(str2,str1,6);           //str2 == "Главный сервер"
strcat(str1,str3);             //str1 == "серверный зал";
```

Функции поиска символа в строке:

```
char * strchr(const char *string, int c);
char * strrchr(const char *string, int c);
```

В первом параметре *string* указывается строка, в которой осуществляется поиск. Во втором параметре *c* указывается символ, который необходимо найти в строке. Функции возвращают указатель на найденный символ или значение **NULL**, если символ не был найден. Функция **strchr** начинает поиск символа с начала строки, а функция **strrchr** - с конца строки. Пример:

```
char str[] = "Строка для поиска";
char *str1 = strchr(str,'к'); //str1 == "ка для поиска"
char *str2 = strrchr(str,'к');//str2 == "ка"
```

Функция поиска строки в строке:

```
char * strstr(const char *str, const char *substr);
```

В первом параметре *str* указывается строка, в которой осуществляется поиск. Во втором параметре *substr* указывается строка, которую необходимо найти. Функция возвращает указатель на первое вхождение строки *substr* в строку *str*. Пример:

```
char str[] = "Строка для поиска";
char *str1 = strstr(str, "для"); //str1 == "для поиска"
```

Функция поиска первого символа в строке из заданного набора символов:

```
size_t strcspn(const char *str, const char *charset);
```

В первом параметре *str* указывается строка, в которой осуществляется поиск символа. Во втором параметре *charset* указывается строка, содержащая набор символов, поиск которых осуществляется. Функция возвращает индекс первого найденного символа из набора *charset* в строке *str*. Символ завершения строки также входит в искомые значения. Пример:

```
char str[] = "Компьютер";
char ch = str[strcspn(str, "трм")]; //ch == 'м'
```

Функции поиска первого символа в строке не принадлежащему заданному набору символов:

```
size_t strspn(const char *str, const char *charset);
```

В первом параметре *str* указывается строка, в которой осуществляется поиск символа. Во втором параметре *charset* указывается строка, содержащая набор символов, поиск которых запрещен. Функция возвращает индекс в строке *str* первого найденного символа, не принадлежащего набору *charset*. Символ завершения строки не включается в искомые значения. Пример:

```
char str[] = "Компьютер";
char ch = str[strspn(str, "Кмъоп")]; //ch == 'ю'
```

Функции поиска первого символа в строке из заданного набора символов:

```
char * strpbrk(const char *str, const char *charset);
```

В первом параметре *str* указывается строка, в которой осуществляется поиск символа. Во втором параметре *charset* указывается строка, содержащая набор символов, поиск которых осуществляется. Функция возвращает указатель на первый найденный символ из набора *charset* в строке *str*. Символ завершения строки не входит в искомые значения. Функция возвращает значение **NULL**, если символ не найден.

Пример:

```
char str[] = "Компьютер";
char *ptr = strpbrk(str, "трм");
char ch = *ptr; //ch == 'м'
```

Функция поиска следующего литерала в строке:

```
char * strtok(char * restrict string,
             const char * restrict charset);
```

В первом параметре *string* указывается строка, в которой осуществляется поиск следующего литерала. Во втором параметре указывается набор возможных символов разделителей литералов. Для кратного поиска литералов в одной строке, строка указывается только при первом вызове функции, в последующих вызовах в первом параметре передается значение **NULL**. Функция возвращает указатель на следующий в строке литерал, или значение **NULL**, если литерал не найден. Например, необходимо подсчитать количество слов в строке. Слова разделяются пробелами или табуляциями. Стока содержится в переменной *str*. Фрагмент программы:

```
char *ptr = strtok(str, "\t");
unsigned num = 0;
while(ptr != NULL) {
    num++;
    ptr = strtok(NULL, "\t");
}
```

В первой строке объявляется указатель *ptr* и устанавливается на первый литерал в строке. Далее объявляется и инициализируется переменная *num*, для подсчета слов в строке. Затем в цикле пока значение указателя не равно **NULL** выполняется:

- переменная счетчик слов *num* увеличивается на единицу;
- указатель *ptr* устанавливается на следующий литерал в строке.

После завершения цикла в переменной *num* содержится число слов в строке.

ЗАДАНИЯ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ

Ввод и вывод строк осуществлять, используя функции **gets** и **puts**. Разделение слов в предложениях осуществляется одним или несколькими пробелами

1-3. Разработать программу согласно варианту задания. Максимальная длина строк не менее 50 символов. При реализации программы необходимо использовать функции для обработки строк из библиотек **stdlib.h** и **string.h**

4. Разработать программу согласно варианту задания. Максимальная длина строк не менее 50 символов. При реализации программы использование функций для работы со строками из библиотек **stdlib.h** и **string.h** запрещено.

5. Разработать программу согласно варианту задания. Максимальная длина строк не менее 80 символов. При реализации программы необходимо использовать функции для обработки строк из библиотек **stdlib.h** и **string.h**

ВАРИАНТЫ ЗАДАНИЙ

| Вариант | Задание |
|----------------|--|
| 1 | <ol style="list-style-type: none">1) Даны строка. Вывести ее три раза через запятую и показать количество символов в ней.2) Даны две строки. Определите, можно ли из некоторых символов первой строки составить вторую строку.3) Удалить в строке все цифры.4) Даны строка предложение. Определить количество слов в строке содержащих две буквы С. Буква С вводится пользователем.5) Задана строка, содержащая целые числа, разделенные пробелами (одним или несколькими). Определить сумму чисел в строке. |
| 2 | <ol style="list-style-type: none">1) Даны строка. Вывести первый, последний и средний (если он есть) символы.2) Даны две строки. Определите, содержится ли меньшая по длине строка в большей.3) Удалите в строке все символы "!".4) Даны строка. Найти символ в строке, встречающийся наибольшее число раз.5) Даны строка содержащая слова. Вывести на экран слово, являющееся последним в алфавитном порядке. |
| 3 | <ol style="list-style-type: none">1) Даны строка. Вывести первые три символа и последний три символа, если длина строки больше 5. Иначе вывести первый символ столько раз, какова длина строки.2) Даны строка. Удалите k-ый символ в ней.3) В строке найдите все серии подряд идущих пробелов и замените каждую на один пробел.4) Даны строка предложение. Преобразовать первые символы всех слов к верхнему регистру.5) Даны строка содержащая слова. Вывести на экран слово, являющееся первым в алфавитном порядке. |
| 4 | <ol style="list-style-type: none">1) Сформировать строку из 10 символов. На четных позициях должны находиться четные цифры, на нечетных позициях - буквы.2) Дан текст. Сформировать строку из символов, расположенных между первой и второй запятыми данного текста.3) В данной строке вставить после каждого символа 'a' символ 'b'.4) Даны строка. Найти символ в строке, встречающийся наименьшее число раз.5) Даны строка, содержащая полное имя файла. Выделить из строки название последнего каталога (без символов "\\"). Если файл содержится в корневом каталоге, то вывести символ "\\". |

| | |
|---|--|
| 5 | <p>1) Даны строка. Показать номера символов, совпадающих с последним символом строки.</p> <p>2) Даны строка. Заменить все символы 'a' и 'b' на 'A' и 'B' соответственно.</p> <p>3) Даны две строки. Удалить в первой строке первое вхождение второй строки.</p> <p>4) Даны строка. Удалить из строки все двойные символы. Пример: "asddewwf" > "asdewf"</p> <p>5) Даны строка предложение и слово. Определить число вхождений заданного слова в предложение.</p> |
| 6 | <p>1) Даны строка. Показать третий, шестой, девятый и так далее символы.</p> <p>2) Дан массив строк. Вывести каждую строку с указанием количества цифр в ней.</p> <p>3) В строке записано десятичное число. Запишите данное число римскими цифрами.</p> <p>4) Даны строка предложение. Определить в предложении слова максимальной и минимальной длины.</p> <p>5) Даны строка, содержащая полное имя файла, то есть имя диска, список каталогов (путь), собственно имя и расширение. Выделить из этой строки расширение файла.</p> |
| 7 | <p>1) Даны строка. Определите общее количество символов '+' и '-' в нем. А также сколько таких символов, после которых следует цифра ноль.</p> <p>2) Дан массив строк. Вывести каждую строку с указанием ее длины.</p> <p>3) Дан email в строке. Определить, является ли он корректным (наличие символа @ и точки, наличие не менее двух символов после последней точки и т.д.).</p> <p>4) Дано целое число. Преобразовать его в строку.</p> <p>5) Задана строка, содержащая целые числа, разделенные пробелами (одним или несколькими). Определить сумму чисел в строке.</p> |
| 8 | <p>1) Даны строка. Определите, какой символ в ней встречается раньше: 'x' или 'w'. Если какого-то из символов нет, вывести сообщение об этом.</p> <p>2) Даны строка. Разделить строку на фрагменты по три подряд идущих символа. В каждом фрагменте средний символ заменить на случайный символ, не совпадающий ни с одним из символов этого фрагмента.</p> <p>3) Дано натуральное число. Получить строку, в которой тройки цифр этого числа разделены пробелом, начиная с правого конца. Например, число 1234567 преобразуется в 1 234 567.</p> <p>4) Даны строка предложение. Определить количество слов в строке содержащих две буквы С. Буква С вводится пользователем.</p> <p>5) Даны строка содержащая слова. Вывести на экран слово, являющееся последним в алфавитном порядке.</p> |

| | |
|----|--|
| 9 | <p>1) Даны две строки. Вывести большую по длине строку столько раз, на сколько символов отличаются строки.</p> <p>2) Даны два слова. Найдите только те символы слов, которые встречаются в обоих словах только один раз.</p> <p>3) Удалить в строке все цифры.</p> <p>4) Данна строка. Найти символ в строке, встречающийся наибольшее число раз.</p> <p>5) Данна строка содержащая слова. Вывести на экран слово, являющееся первым в алфавитном порядке.</p> |
| 10 | <p>1) Данна строка. Если она начинается на 'abc', то заменить их на 'www', иначе добавить в конец строки 'zzz'.</p> <p>2) Дан текст. Найдите наибольшее количество подряд идущих пробелов в нем.</p> <p>3) Удалите в строке все символы "!".</p> <p>4) Данна строка предложение. Преобразовать первые символы всех слов к верхнему регистру.</p> <p>5) Данна строка, содержащая полное имя файла. Выделить из строки название последнего каталога (без символов "\\"). Если файл содержится в корневом каталоге, то вывести символ "\\".</p> |
| 11 | <p>1) Данна строка. Если ее длина больше 10, то оставить в строке только первые 6 символов, иначе дополнить строку символами 'o' до длины 12.</p> <p>2) Дан текст. Найти слова, состоящие из цифр, и сумму чисел, которые образуют эти слова.</p> <p>3) В строке найдите все серии подряд идущих пробелов и замените каждую на один пробел.</p> <p>4) Данна строка. Найти символ в строке, встречающийся наименьшее число раз.</p> <p>5) Данна строка предложение и слово. Определить число вхождений заданного слова в предложение.</p> |
| 12 | <p>1) В данной строке найти количество цифр.</p> <p>2) Дан текст. Найти сумму имеющихся в нем цифр.</p> <p>3) В данной строке вставить после каждого символа 'a' символ 'b'.</p> <p>4) Данна строка. Удалить из строки все двойные символы. Пример: "asddewwf" > "asdewf"</p> <p>5) Данна строка, содержащая полное имя файла, то есть имя диска, список каталогов (путь), собственно имя и расширение. Выделить из этой строки расширение файла.</p> |
| 13 | <p>1) Данна строка. Определить, содержит ли строка только символы 'a', 'b', 'c' или нет.</p> <p>2) Дан текст. Найдите наибольшее количество идущих подряд цифр.</p> <p>3) Даны две строки. Удалить в первой строке первое вхождение второй строки.</p> |

| | |
|----|--|
| | <p>4) Дано строка предложение. Определить в предложении слова максимальной и минимальной длины.</p> <p>5) Задана строка, содержащая целые числа, разделенные пробелами (одним или несколькими). Определить сумму чисел в строке.</p> |
| 14 | <p>1) Данна строка. Вывести ее три раза через запятую и показать количество символов в ней.</p> <p>2) Данна строка, состоящая из слов, разделенных символами, которые перечислены во второй строке. Показать все слова.</p> <p>3) В строке записано десятичное число. Запишите данное число римскими цифрами.</p> <p>4) Дано целое число. Преобразовать его в строку.</p> <p>5) Данна строка содержащая слова. Вывести на экран слово, являющееся последним в алфавитном порядке.</p> |
| 15 | <p>1) Данна строка. Вывести первый, последний и средний (если он есть) символы.</p> <p>2) Удалить в строке все лишние пробелы, то есть серии подряд идущих пробелов заменить на одиночные пробелы. Крайние пробелы в строке удалить.</p> <p>3) Дан email в строке. Определить, является ли он корректным (наличие символа @ и точки, наличие не менее двух символов после последней точки и т.д.).</p> <p>4) Данна строка предложение. Определить количество слов в строке содержащих две буквы С. Буква С вводится пользователем.</p> <p>5) Данна строка содержащая слова. Вывести на экран слово, являющееся первым в алфавитном порядке.</p> |
| 16 | <p>1) Данна строка. Вывести первые три символа и последний три символа, если длина строки больше 5. Иначе вывести первый символ столько раз, какова длина строки.</p> <p>2) Найдите количество вхождения 'aba' в строку.</p> <p>3) Дано натуральное число. Получить строку, в которой тройки цифр этого числа разделены пробелом, начиная с правого конца. Например, число 1234567 преобразуется в 1 234 567.</p> <p>4) Данна строка. Найти символ в строке, встречающийся наибольшее число раз.</p> <p>5) Данна строка, содержащая полное имя файла. Выделить из строки название последнего каталога (без символов "\\"). Если файл содержится в корневом каталоге, то вывести символ "\\".</p> |
| 17 | <p>1) Сформировать строку из 10 символов. На четных позициях должны находиться четные цифры, на нечетных позициях - буквы.</p> <p>2) Удалите в строке все 'abc', за которыми следует цифра.</p> <p>3) Удалить в строке все цифры.</p> <p>4) Данна строка предложение. Преобразовать первые символы всех слов к верхнему регистру.</p> |

| | |
|----|--|
| | 5) Дано строка предложение и слово. Определить число вхождений заданного слова в предложение. |
| 18 | <p>1) Дано строка. Показать номера символов, совпадающих с последним символом строки.</p> <p>2) Удалите в строке все буквы 'x'. за которыми следует 'abc'.</p> <p>3) Удалите в строке все символы "!".</p> <p>4) Дано строка. Найти символ в строке, встречающийся наименьшее число раз.</p> <p>5) Дано строка, содержащая полное имя файла, то есть имя диска, список каталогов (путь), собственно имя и расширение. Выделить из этой строки расширение файла.</p> |
| 19 | <p>1) Дано строка. Показать третий, шестой, девятый и так далее символы.</p> <p>2) Замените в строке все вхождения 'word' на 'letter'.</p> <p>3) В строке найдите все серии подряд идущих пробелов и замените каждую на один пробел.</p> <p>4) Дано строка. Удалить из строки все двойные символы. Пример: "asddewwf" > "asdewf"</p> <p>5) Задана строка, содержащая целые числа, разделенные пробелами (одним или несколькими). Определить сумму чисел в строке.</p> |