ТИПЫ ДАННЫХ. ОПЕРАТОРЫ. ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

Всякая программа есть последовательность операторов, которые подразделяются на простые и структурные. Каждый оператор имеет определенную структуру и записывается с использованием служебных слов и символов языка.

Синтаксис оператора есть правило его описания, которое может быть задано либо в виде общей формы записи оператора, либо в виде синтаксической диаграммы. Синтаксическая диаграмма помимо синтаксиса задает и семантику оператора, т.е. определяет те действия, которые заложены в этом операторе, и порядок выполнения этих действий. Для некоторых сложных операторов помимо синтаксической диаграммы необходимо давать дополнительные пояснения по их семантике.

Различают простые и структурные операторы. *Простым* оператором является оператор, не содержащий в себе других операторов. В простом операторе определяется, как правило, одно элементарное действие. В Паскале имеются три простых оператора: присваивания, вызова процедуры и перехода. *Структурные* операторы подразделяются, в свою очередь, на составные, условные, цикла и операторы над записями. Структурный оператор включает в себя другие операторы (как простые, так и составные).

Типы данных

Целочисленный тип

Сюда входят несколько целочисленных типов, которые различаются диапазоном значений, количеством байт отведённых для их хранения и словом, с помощью которого объявляется тип.

Тип	Диапазон	Размер в байтах
shortint	-128…127	1
integer	-32 768…32 767	2
longint	-2 147 483 648…2 147 483 647	4
byte	0…255	1
word	0…65 535	2

Объявить целочисленную переменную можно в разделе Var, например:

Var book: word;

Над переменными этой категории можно выполнять все арифметические и логические операции за исключением деления (/), для него нужен вещественный тип. Также могут быть применены некоторые стандартные функции и процедуры.

Вещественный тип

В Паскале бывают следующие вещественные типы данных:

Тип	Диапазон	Память, байт	Количество цифр
Real	2.9e-39 … 1.7e38	6	11-12
Single	1.5e-45 … 3.4e38	4	7-8
Double	5.0e-324 …1.7e308	8	15-16
Extended	3.4e-4932 … 1.1e493	10	19-20
Comp	-9.2e63 … (9.2e63)-1	8	19-20

Над ними может быть выполнено большее количество операций и функций, чем над целыми. Например, эти функции возвращают вещественный результат:

sin(x) – синус;

cos(x) – косинус;

arctan(x) – арктангенс;

ln(x) – натуральный логарифм;

sqrt(x) – квадратный корень;

exp(x) – экспонента;

Логический тип

Переменная, имеющая логический тип данных может принимать всего два значения: true (истина) и false (ложь). Здесь истине соответствует значение 1, а ложь тождественная нулю. Объявить булеву переменную можно так:

Var A: Boolean;

Над данными этого типа могут выполняться операции сравнения и логические операции: not , and, or, xor.

Символьный тип

Символьный тип данных – это совокупность символов, используемых в том или ином компьютере. Переменная данного типа принимает значение одного из этих символов, занимает в памяти компьютера 1 байт. Слово **Char** определяет величину данного типа. Существует несколько способов записать символьную переменную (или константу):

- как одиночный символ, заключенный в апострофы: ‘W’, ‘V’, ‘п’;
- указав код символа, значение которого должно находиться в диапазоне от 0 до 255.
- при помощи конструкции ^K, где K – код управляющего символа. Значение K должно быть на 64 больше кода соответствующего управляющего символа.

К величинам символьного типа данных применимы операции отношения и следующие функции:

Succ(x) — возвращает следующий символ;

Pred(x) — возвращает предыдущий символ;

Ord(x) — возвращает значение кода символа;

Chr(x) — возвращает значение символа по его коду;

UpCase(x) — переводит литеры из интервала ‘a’..’z’ в верхний регистр.

Для плодотворной работы с символьным типом рекомендую пользоваться таблицей ASCII.

Строковый тип

Строка в Паскале представляет собой последовательность символов заключенных в апострофы, и обозначается словом **String**. Число символов (длина строки) должно не превышать 255. Если длину строки не указывать, то она автоматически определиться в 255 символов. Общий вид объявления строковой переменной выглядит так:

Var <имя_переменной>: string[<длина строки>];

Каждый символ в строке имеет свой индекс (номер). Индекс первого байта – 0, но в нем храниться не первый символ, а длина всей строки, из чего следует, что переменная этого типа будет занимать на 1 байт больше числа переменных в ней. Номер первого символа – 1, например, если мы имеем строку S=‘stroka’, то S[1]=s;. В одном из следующих уроков строковый тип данных будет рассмотрен подробнее.

Перечисляемый тип данных

Перечисляемый тип данных представляет собой некоторое ограниченное количество идентификаторов. Эти идентификаторы заключаются в круглые скобки, и отделяются друг от друга запятыми.

Пример:

Type Day=(Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday);

Var A: Day;

Переменная A может принимать лишь значения определенные в разделе Type. Также можно объявить переменную перечисляемого типа в разделе Var:

Var A: (Monday, Tuesday);

К данному типу применимы операции отношения, при этом заранее определенно, что Monday<Tuesday<Wednesday т. д. Также можно применять функции succ, pred, ord, процедуры inc и dec, и использовать операцию присваивания: A:=Tuesday;

Интервальный тип данных

Когда необходимо задать какой то диапазон значений, то в таких ситуациях применяется интервальный тип данных. Для объявления используется конструкция **m..n**, где **m** – минимальное (начальное) значение, а **n** – максимально (конечное); здесь m и n являются константами, которые могут быть целого, символьного, перечисляемого или логического типа.

Описываться величины интервального типа могут как в разделе типов, так и в разделе описания переменных.

Общий вид:

TYPE <имя_типа> = <мин. значение>..<макс. значение>;

Пример:

TYPE Cards = 1..36;

Оператор присваивания и выражения

Оператор присваивания относится к простым операторам. По этому оператору переменной присваивается значение выражения. Несмотря на кажущуюся простоту оператора присваивания, при его выполнении осуществляется целый набор элементарных действий:

- переменные, находящиеся в выражении, получают свои значения;
- вычисляется значение выражения;
- переменной слева от знака присваивания «:=» присваивается полученное значение.

В простейшем случае, когда выражение задано константой или другой переменной, вычислений не производится и переменная сразу получает свое значение, например:

```
RAZN := A – 3.5;
N := 25; C := D;  Y := 'программа';
L := true; P := X > 10.
```

В языке Паскаль существует несколько типов выражений: арифметические, литерные, логические (булевские). В этом пункте мы рассмотрим только арифметические выражения.

Арифметические выражения складываются из констант, переменных, стандартных функций с использованием скобок и знаков операций. В Паскале определены следующие операции над числами: * , / , + , – , DIV, MOD, где DIV – деление нацело; MOD – вычисление остатка от деления. Приоритеты:

```
*, /, DIV, MOD – высший;
+, –           – низший.
```

Например:

A := 13 DIV 5; (результат: A = 2),

B := 13 MOD 5; (результат: B = 3).

Каждое арифметическое выражение может иметь типы INTEGER и REAL. Тип константы определяется самим видом константы, тип переменной задается в ее объявлении.

Тип арифметического выражения определяется по следующему правилу:

1. Для операций «*, +, -» результат имеет тип REAL, если хотя бы один из операндов имеет тип REAL. Если оба операнда типа INTEGER, то и результат имеет тип INTEGER.

2. Для «/» результат всегда имеет тип REAL.

3. Для «DIV, MOD» операнды и результат имеют тип INTEGER.

Значение переменной интервального типа, образованной на основе INTEGER, всегда имеет тип INTEGER. При использовании оператора присваивания нужно соблюдать типизацию объектов слева и справа от знака «:=». Смешение типов недопустимо за исключением случая, когда слева от знака «:=» стоит тип REAL, а справа – тип INTEGER.

Операторы вызова процедур. Ввод/вывод данных

Оператор вызова процедуры определяет активизацию процедуры, обозначенную с помощью идентификатора (имени) процедуры. Другими словами, с помощью операторов этого типа осуществляется вызов процедур с указанием в них входных и выходных параметров (подробнее об этом будет сказано в разделе «Процедуры»). Мы начнем знакомство с операторами-процедурами на базе организации ввода/вывода данных в языке Паскаль.

Для организации ввода и вывода данных используются следующие встроенные (машинные) процедуры: READ, WRITE, READLN, WRITELN.

Процедуры ввода READ и READLN

Процедура READ вызывается с помощью соответствующего оператора процедуры.

Общая форма записи оператора

READ (X,Y, ... , Z), где X,Y, ... , Z – переменные, называемые списком ввода.

При выполнении процедуры READ работа программы приостанавливается, ЭВМ ждет ввода данных. Пользователь должен с клавиатуры ввести значения переменных, указанных в списке, отделяя их одним пробелом. Ввод завершается нажатием клавиши ENTER. Можно нажимать клавишу ввода и после набора каждого элемента ввода. В этом случае каждое нажатие клавиши ENTER осуществляет присваивание очередной переменной списка ввода ее значения, набранного с клавиатуры. По завершении ввода программа возобновляет свою работу.

Для лучшего понимания работы данной процедуры и ее умелого использования при задании значений нескольких переменных необходимо знать, что при вводе значений переменных (констант) с клавиатуры они сначала идут в буфер клавиатуры, а потом считываются в ячейки оперативной памяти, отведенные компилятором этим переменным. При считывании буфер очищается по принципу очереди (первым зашел – первым вышел). Это означает, что при вводе сразу нескольких констант и при последующем нажатии клавиши ENTER из буфера клавиатуры будет считано столько констант, сколько переменных в операторе READ, а остальные останутся в буфере. Если же в буфере клавиатуры после очередного считывания останутся еще константы, то при следующем операторе READ остановки работы программы не будет, и его переменные получат свои

значения из буфера (если только в нем достаточно констант для всех переменных).

Например, пусть имеется фрагмент программы, включающий в себя два оператора READ:

.....
READ (A, B, C);
……
READ (D, E);

и пусть по первому оператору READ на клавиатуре набрано 5 констант. Тогда при работе второго READ останова работы программы не будет и переменные C и D получат значения последних двух ранее введенных констант. Если же ввести 4 константы, то второй оператор READ затребует еще одну константу с клавиатуры.

Вызов процедуры READLN имеет тот же синтаксис, что и оператор READ, однако ее работа отличается от работы первой процедуры. При однократном вводе констант отличий нет, а при одноразовом вводе нескольких констант происходит очистка буфера клавиатуры. Так, если в нашем примере заменить первый READ на READLN и тоже ввести сразу 5 констант, то второй оператор READ произведет остановку работы программы и затребует повторного ввода последних двух значений для переменных D и E. Заметим также, что оператор READLN используется преимущественно при вводе текстовых констант (READLN – read line – читать текст).

Процедуры вывода WRITE и WRITELN

Процедуры вывода WRITE и WRITELN служат для вывода на экран констант (как числовых, так и текстовых), значений переменных и выражений. Они вызываются с помощью одноименных операторов вызова процедур, например:

WRITE ('программа', X, Y – Z * 3).

По этому оператору на экран будет выведено в одной строке слово «программа» и далее без пробелов значения переменной X и выражения Y – Z * 3. Например, если имеем X = -3, Y = -5, Z = 12, то на экран будет выведено: программа-3-41.

Чтобы отделить элементы вывода друг от друга, используется прием форматирования вывода. Так, WRITE (A:20) – одиночное форматирование – показывает, что значению переменной A отводится 20 позиций на экране монитора. Если в значение переменной A входит менее 20 символов, то они сдвигаются вправо, а слева строка заполняется пробелами.

Двойное форматирование используется только для вывода вещественных значений. Например, WRITE (C:17:7) означает, что для вывода значения переменной C отведено всего 17 позиций, из них 7 позиций предназначены для представления дробной части. Если формат не указан, то вещественные константы выводятся на экран в экспоненциальной форме.

Заметим также, что форматировать в операторах WRITE можно не только переменные, но и выражения, например:

WRITE (cos (x + 4) : 5 : 2);

Работа оператора WRITE отличается от работы оператора WRITELN тем, что по завершении вывода у WRITE курсор остается в конце списка вывода, а у WRITELN он переходит на следующую строку. Часто используют оператор WRITELN без списка вывода для вывода на экран пустой строки.

Проиллюстрируем работу этих операторов на следующем примере:

```
program AVERAGE;
    var FIRST, SECOND, TROIS, SUM: integer;
       begin
writeln ('Введите 3 числа ');
readln (FIRST, SECOND, TROIS);
SUM := FIRST + SECOND + TROIS;
writeln ('Среднее значение ', FIRST:4,',',SECOND:4,',');
write (TROIS:4,'  равно ';(SUM div 3):3)
end.
```

На экран будет выведено:

```
Введите 3 числа
2  12  9
Среднее значение    3,    12,
9  равно   8
```

Основные стандартные процедуры и функции

Для построения вычисляемых выражений, используемых в правой части оператора присваивания или в операторах вывода, можно применить ряд стандартных встроенных функций. Большинство из них (таблица 1) имеют в качестве аргумента (аргументов) данные вещественных и целых типов, некоторые − только вещественных.

Используя в программном коде стандартные функции, следует помнить, что **аргумент всегда надо брать в круглые скобки**!

Таблица 1 − **Стандартные математические функции**

Запись функции на Pascal	Математическая запись	Тип аргумента	Тип результата	Действие
abs(*x*)	$\|x\|$	integer, real	integer, real	модуль (абсолютное значение) числа *x*

sqr(*x*)	x^2	integer, real	integer, real	квадрат числа *x*
sqrt(*x*)	$\sqrt{x}, x \geq 0$	integer, real	real	квадратный корень из числа *x*
sin(*x*)	$\sin x$	integer, real	real	синус числа *x*
cos(*x*)	$\cos x$	integer, real	real	косинус числа *x*
arctan(*x*)	arctgx	integer, real	real	арктангенс числа *x*
ln(*x*)	$\ln x, x > 0$	integer, real	real	натуральный логарифм числа*x*
exp(*x*)	e^x	integer, real	real	экспонента числа *x*, т. е. $2{,}718^x$
trunc(*x*)	–	real	integer	отбрасывает дробную часть *x*
round(*x*)	–	real	integer	округляет число *x*
int(*x*)	–	real	real	целая часть числа *x*
frac(*x*)	–	real	real	дробная часть числа *x*

Линейным называется алгоритм, в котором результат получается путем однократного выполнения заданной последовательности действий при любых значениях исходных данных. Операторы программы выполняются последовательно, один за другим, в соответствии с их расположением в программе.

Пример. Определить расстояние на плоскости между двумя точками с заданными координатами $M_1(x_1,y_1)$ и $M_2(x_2,y_2)$.

Решение задачи.

В этом примере проведем полный разбор решения задачи.

Математическая модель: расстояние на плоскости между двумя точками $M_1(x_1,y_1)$ и $M_2(x_2,y_2)$ высчитывается по формуле:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Составим блок-схему алгоритма, а затем уточним содержимое блоков "Вычисление расстояния" и "Вывод расстояния" (см. рис.1):

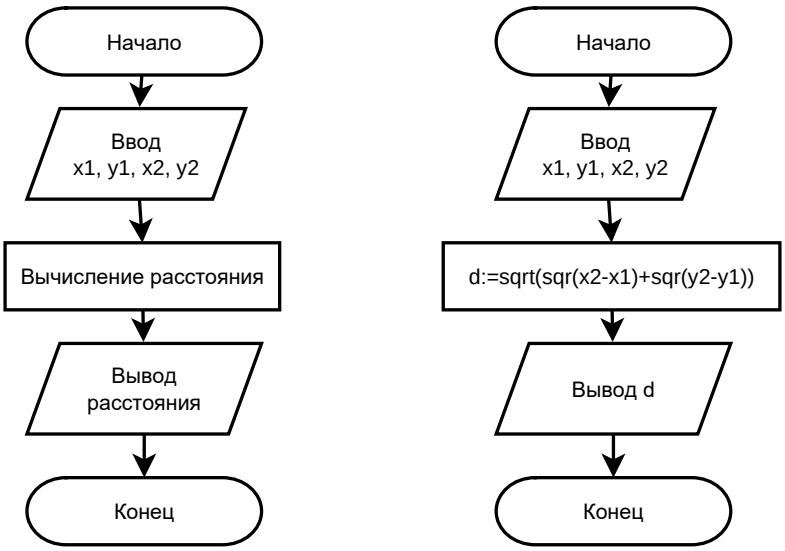

Рис. 1. Блок-схема алгоритма

Дальнейшая детализация не требуется. Переводим блок-схему на язык Паскаль, доработав программу, чтобы улучшить ее интерфейс:

```
program example1;
var x1, x2, y1, y2: Integer;
   d:Real;
begin
   Writeln('Эта программа вычисляет расстояние между двумя точками на плоскости');
   Writeln('Введите координаты двух точек:');
   Write('x1= '); Readln(x1);
   Write('y1= '); Readln(y1);
   Write('x2= '); Readln(x2);
   Write('y2 ='); Readln(y2);
   d:=sqrt(sqr(x2-x1)+sqr(y2-y1));
   Writeln('d= ',d:6:2);
   Writeln('нажмите Enter для завершения работы программы');
   Readln;
end.
```

Вопросы для самопроверки:

1. Какой оператор называется простым?
2. Какой оператор называется структурным?
3. Какие существуют типы данных?
4. Как обозначается начало и конец программы?
5. Из каких разделов состоит программа на языке Паскаль?
6. Как в языке Паскаль осуществляется ввод/вывод данных?
7. Для чего предназначен оператор присваивания?

Задания

1. Три сопротивления R1, R2, R3 соединены параллельно. Найти сопротивление соединения.
2. Определить время падения камня на поверхность земли с высоты h.
3. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.
4. Вычислить высоту треугольника, опущенную на сторону a, по известным значениям длин его сторон a, b, c.
5. Вычислить объем цилиндра с радиусом основания r и высотой h.
6. Определить расстояние, пройденное физическим телом за время t, если тело движется с постоянным ускорением a и имеет в начальный момент времени скорость V0.
7. Вычислить площадь треугольника по формуле Герона, если заданы его стороны.
8. Определить координаты вершины параболы y=ax2+bx+c (a<>0). Коэффициенты a,b,c заданы.
9. По данным сторонам прямоугольника вычислить его периметр, площадь и длину диагонали.
10. Даны два числа. Найти среднее арифметическое их квадратов и среднее арифметическое их модулей.
11. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.
12. Найти длину окружности и площадь круга заданного радиуса R.
13. Даны координаты трех вершин треугольника (x1, y1), (x2, y2), (x3, y3). Найти его периметр и площадь.
14. Дано целое четырехзначное число. Используя операции div и mod, найти сумму его цифр.
15. Дано целое четырехзначное число. Используя операции div и mod, найти произведение его цифр.
16. Скорость первого автомобиля V1 км/ч, второго — V2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили первоначально движутся навстречу друг другу.
17. Скорость лодки в стоячей воде V км/ч, скорость течения реки U км/ч (U < V). Время движения лодки по озеру T1 ч, а по реке (против течения) — T2 ч. Определить путь S, пройденный лодкой.
18. Дана сторона равностороннего треугольника. Найти площадь этого треугольника и радиусы вписанной и описанной окружностей.
19. Известно количество жителей в государстве и площадь его территории. Определить плотность населения в этом государстве.
20. Найти площадь кольца, внутренний радиус которого равен R1, а внешний радиус равен R2 (R1 < R2).