1. Язык программирования высокого уровня PascalABC

Система предназначена для обучения программированию на языке Паскаль и ориентирована на студентов младших курсов. Эта система призвана осуществить переход от простейших программ к модульному, объектно-ориентированному, событийному и компонентному программированию. Многие концепции в Pascal ABC упрощены, что позволяет использовать их на более ранних этапах обучения. Модуль графики обходится без объектов, хотя его возможности практически совпадают с графическими возможностями Borland Delphi.

В консольных программах можно создавать таймеры и звуки, которые реализованы без использования объектов. В модулях может отсутствовать разделение на секцию интерфейса и секцию реализации; в этом случае модули устроены практически так же, как и основная программа, что проще на ранних этапах обучения.

Компилятор Pascal ABC не генерирует исполняемый код в виде .exe-файла, а создает в результате компиляции дерево программы в памяти, которое затем выполняется с помощью встроенного интерпретатора.

Стандартные модули Pascal ABC:

- Модуль растровой графики GraphABC
- Модуль векторной графики ABCObjects для быстрого изучения основ объектно-ориентированного программирования
- Модули Sounds, Timers, Events для работы со звуками, таймерами, событиями
- Модуль Containers, реализующий наиболее распространённые контейнерные классы
- Модули исполнителей Робот и Чертёжник для быстрого обучения основам программирования школьников младших и средних классов

Алфавит языка

Программа на языке PascalABC формируется с помощью конечного набора знаков, образующих *алфавит языка*, и состоит из:

- прописных и строчных букв латинского алфавита (A, B,...,Z, a, b,...,z) и знака подчеркивания;
- цифр (0, 1,...,9)

Кроме того, в алфавит включаются специальные символы и составные символы.

СПЕЦИАЛЬНЫЕ СИМВОЛЫ

Символ	Название	Символ	Название
+	Плюс	{}	Фигурные скобки
-	Минус	.	Точка
*	Звездочка	,	Запятая
/	Дробная черта	:	Двоеточие
=	Равно	;	Точка с запятой
>	Больше	'	Апостроф
<	Меньше	#	Номер
[ ]	Квадратные скобки	$	Знак денежной единицы

()	Круглые скобки	^	Тильда (каре)
@	Коммерческое а		Пробел (не имеет обозначения)

СОСТАВНЫЕ СИМВОЛЫ

Символ	Название	Символ	Название
:=	Присваивание	<=	Меньше или равно
<>	Не равно	>=	Больше или равно
..	Диапазон значений	(. .)	Альтернатива []
(* *)	Альтернатива {}		

Неделимые последовательности знаков алфавита образуют слова, отделенные друг другу разделителями. Разделителями служат - пробел, символ конца строки, комментарий. Пробел, стоящий внутри строковой константы, воспринимается не как разделитель, а как ее часть. Между комбинациями специальных символов пробелы недопустимы.

Слова подразделяются на зарезервированные слова, стандартные идентификаторы (имена) и идентификаторы пользователя.

Зарезервированные слова языка являются составной частью языка, имеют фиксированное начертание и несут в программе определенный смысл.

Идентификаторы и служебные слова

Идентификаторы служат в качестве имен программ, модулей, процедур, функций, типов, переменных и констант. Идентификатором считается любая последовательность латинских букв или цифр, начинающаяся с буквы. Буквой считается также символ подчеркивания "_".

Например, `a1`, `_h`, `b123` - идентификаторы, а `1a`, `ф2` - нет.

Служебные слова служат для оформления конструкций языка и не могут быть использованы в качестве имен. Список всех служебных слов языка **Pascal ABC** приведен в таблице _____:

Таблица _____

```
and          array      as           begin
break        case       class        const
constructor  continue   destructor   div
do           downto     else         end
exit         external   externalsync file
finalization for        forward      function
if           in         inherited    initialization
is           mod        not          of
or           private    procedure    program
property     protected  public
record       repeat     set          shl
shr          sizeof     string
then         to         type         unit
until        uses       var          while
with         xor
```

Общие стандартные функции

Имя и **Тип возвращаемого значения**

параметры

`Abs(x)`	возвращает абсолютное значение (модуль) `x`
`Sqr(x)`	возвращает квадрат `x`
`Sqrt(x)`	возвращает квадратный корень из `x`
`Sin(x)`	возвращает синус `x`
`Cos(x)`	возвращает косинус `x`
`Ln(x)`	возвращает натуральный логарифм `x`
`Exp(x)`	возвращает `e` в степени `x` (`e`=2.718281...)
`Arctan(x)`	возвращает арктангенс x
`Power(x,y)`	возвращает x в степени y
`Random(x)`	возвращает `x` в степени `y`

Приоритет операций

Приоритет определяет порядок выполнения операций в выражении. Первыми выполняются операции, имеющие высший приоритет. Операции, имеющие одинаковый приоритет, выполняются слева направо.

Таблица приоритетов операций

@, not,Л	1 (наивысший)
*, /, div, mod, and, shl, shr	2
+, -, or, xor	3
=, <>, <, >, <=, >=, in	4 (низший)

Типы данных

К основным типам данных языка Pascal относятся:

- целые числа (integer и др.);
- действительные (вещественные) числа (real и др.);
- символы (char);
- строки (string);
- логический (boolean);
- тип "массив";
- тип "запись";
- процедурный и др.

Тип целые числа указываются в разделе описания переменных:

```
var a: integer;
```

Название	Диапазон значений	Память, байт
byte	0-255	1
word	0-65 535	2
integer	-2 147 483 648-2 147 483 647	4

Тип `real` (вещественный). Значения вещественного типа занимают 8 байт, содержат 15-16 значащих цифр и по модулю не могут превосходить величины $1.7 \cdot 10^{308}$. Константы типа `real` можно записывать как в форме с плавающей точкой, так и в экспоненциальной форме: `1.7`, `0.013`, `2.5e3` (2500), `1.4e-1` (0.14).

```
var b: real;
```

Логический (булевский) тип имеет всего два значения: **true** (да - истина, 1) и **false** (нет - ложь, 0), причем данные значения упорядочены, т.е. в операциях сравнения true > false.

```
var L: boolean;
```

Символьный тип занимает 1 байт

```
var c: chr;
```

Строковый тип - строки в **PascalABC** имеют тип **string** и состоят из не более чем 255 символов. При описании может указываться длина строки.

```
var s: string;
```

Для экономии памяти предусмотрено описание вида

```
var s1: string[40];
```

2. Структура программы на языке PASCAL ABC

Программа реализует алгоритм решения задачи. В ней программист записывает последовательность действий, выполняемых над определенными данными с помощью определенных операций для реализации заданной цели. Основные характеристики программы: точность полученного результата, время выполнения и объем требуемой памяти.

Программа на языке **Pascal ABC** имеет следующий вид:

```
program имя программы;
раздел подключения модулей
раздел описаний
begin
…
операторы
…
end.
```

Заголовок программы несет чисто смысловую нагрузку и может отсутствовать, однако рекомендуется всегда его записывать (на латинском регистре) для быстрого распознавания нужной программы среди листингов других программ.

Раздел подключения модулей начинается со служебного слова **uses**, за которым следует список имен модулей, состоящий в общем случае из семи разделов, перечисляемых через запятую:

- описания меток;
- описания констант;
- определения типов данных;
- описания переменных;
- описания процедур и функций;
- операторов.

Любой раздел, кроме раздела операторов, может отсутствовать. Разделы описаний (кроме uses, который всегда расположен после заголовка программы) могут встречаться в программе любое количество раз и следовать в произвольном порядке. Главное, чтобы все описания объектов программы были сделаны до того, как они будут использованы.

Каждая встречающаяся в программе переменная должна быть описана. Описание обязательно предшествует использованию переменной. Раздел описания переменных начинается зарезервированным словом **var** (variable — переменная), затем через запятую перечисляются имена переменных и через двоеточие следуют их тип и точка с запятой. Формат:

var
<идентификатор, . . . > : <тип>;

В рассматриваемом примере программы три переменных A, B и Proizved могут принимать целочисленные значения, описаны следующим образом:

```
var
A,B, Proizved : integer;
```

Раздел описания процедур и функций

В общем случае подпрограмма имеет ту же структуру, что и программа. Для описания подпрограмм используются зарезервированные слова **procedure и function,** которые записываются в начале подпрограммы. Формат процедуры:

procedure <имя процедуры> {<параметры>} ;
<разделы описаний>
<раздел операторов>
end;

Формат функции:

function <имя функции> {<параметры>} : <тип результата>;
<разделы описаний>
<раздел операторов>
end;

Раздел операторов

В программе на языке Паскаль раздел операторов является основным, так как именно в нем с предварительно описанными переменными, константами, значениями функций выполняются действия, позволяющие получить результат, ради которого создавалась программа.

Раздел операторов начинается зарезервированным словом **begin** (начало), далее следуют операторы языка, отделенные друг от друга точкой с запятой. Завершает раздел зарезервированное слово end (конец) с точкой.

Например:

```
begin                    {Начало программы}
Write ('Введите значение целого числа A >');{Вывод запроса на экран}
Readln (A);              {Ввод значения A с клавиатуры}
<операторы>
…
end.                     {Конец программы}
```

Операторы выполняются строго последовательно в том порядке, в котором они записаны в тексте программы в соответствии с синтаксисом и правилами пунктуации.

Слова **begin и end** являются аналогом открывающей и закрывающей скобки в обычных арифметических выражениях.

Комментарии

Для лучшего понимания программы в ней записывается пояснительный текст — **комментарий.** Комментарий можно записать в любом месте программы, где разрешен пробел. Текст комментария ограничен символами { } или (* *) и может содержать любые комбинации латинских и русских букв, цифр и других символов алфавита языка Паскаль. Ограничений на длину комментария нет, он может занимать несколько строк.

Примеры.

// Начало программы

{Начало программы} или (*Начало программы*)

В ограничителях (* *) пробелы между скобкой и звездочкой запрещены. В тексте не должны находиться знаки ограничителей, с которых комментарий начинается. Комментарий игнорируется компилятором и поэтому никакого влияния на программу не оказывает. По месту положения в программе комментарии можно подразделить на четыре класса: объясняющие назначение программы, поясняющие смысл идентификаторов переменных и констант, описывающие логически обособленные части программы, объясняющие трудно понимаемые элементы алгоритма. В удачно прокомментированной программе легко найти ошибку, проанализировав различие между замыслом автора (в комментариях) и реализацией (в тексте программы).

Ограничители { } и (* *) удобно использовать при отладке программ. В процессе отладки часто требуется временно исключить выполнение какой-либо части программы. Конечно, этого можно добиться, уничтожив временно ненужные операторы или обойдя их с помощью оператора go to. Однако оба этих способа неприемлемы по ряду совершенно понятных причин: повторный набор вновь понадобившихся операторов, путаница с операторами go to и т.д. Гораздо удобнее просто заключить временно ненужную часть программы в {} или (* *), которая будет восприниматься компилятором как комментарий.

Например:

```
begin                    {Начало программы}
Write ('Введите значение целого числа A >'); {Вывод запроса на экран}
Readln (A);              {Ввод значения A с клавиатуры}
C:= A * B;       {Вычисление переменной C}
{Временно невыполняемая часть программы
end.                     {Конец программы}
```

При необходимости { } или (* *) можно убрать, и программа будет выполняться в полном объеме.

3. Работа в интегрированной оболочке PascalABC

Ввод и редактирование программы

В качестве редактора для системы **Pascal ABC** использован компонент **SynEdit.**

Набор строки заканчивается нажатием клавиши Enter для перехода указателя на новую строку.

Вставка символа - подвести указатель мыши на нужное место и набрать недостающие символы.

Удаление символа. Подвести указатель на удаляемый символ:

- нажатием клавиши **Del** - удаляется выбранный символ, а строка сжимается справа к указателю мыши;

- нажатием клавиши **Back Space** - удаляется символ стоящий слева от указателя, а строка сдвигается влево от указателя мыши.

Вставка строки. Маркер установить на конец строки, после которой вставить пустую (или на начало перед которой вставить пустую) и нажать **Enter**.

Удаление строки. Маркер установить на нужную строку и нажать клавиши **CTRL + Y**.

Работа с фрагментом программы:

- выделить фрагмент, для копирования;

- скопировать выделенный фрагмент в буфер обмена выбрав пункты меню **ПРАВКА, КОПИРОВАТЬ;**

- установить указатель мыши на место вставки фрагмента

- вставить фрагмент из буфера обмена выбрав пункты меню **ПРАВКА, ВСТАВИТЬ.**

Сохранение файла. Для сохранения файла выбрать из меню команду **Файл Сохранить** или нажать клавиши **Ctrl + S** (при первом сохранении ввести имя файла в котором будет сохранена программа).

Горячие клавиши при работе с редактором

F2, **Ctrl-S** - сохранить файл.

F3, **Ctrl-O** - загрузить файл.

F12 - сохранить файл под новым именем.

Ctrl-Shift-S - сохранить все открытые файлы.

Ctrl-Shift-0 ... **Ctrl-Shift-9** - установить закладку с номером 0...9.

Ctrl-0 ... **Ctrl-9** - перейти к закладке с номером 0...9.

Ctrl-Tab, **Ctrl-Shift-Tab** - перейти к следующему/предыдущему окну редактора.

Ctrl-Shift-I - увеличить отступ выделенного блока.

Ctrl-Shift-U - уменьшить отступ выделенного блока.

Окна интерфейса:

- окно редактора;
- окно вывода;
- окно ввода;
- окно выполненной программы;
- окно отладки.

В окне **редактора** обеспечивается набор и редактирование программы.

Под окном редактора расположено окно вывода. Оно предназначено для вывода данных процедурами `write` и `writeln`, а также для вывода сообщений об ошибках и предупреждений во время работы программы.

Окно вывода может быть скрыто. Клавиша **F5** и кнопка **ОКНО ВЫВОДА(F5)** показывают/скрывают окно вывода. Для скрытия окна вывода используется также клавиша **Esc**.

Окно **вывода** обязательно открывается при любом выводе в него.

Для очистки окна вывода следует нажать комбинацию клавиш **Ctrl-Del** или кнопку .

Окно **ввода** открывается при выполнении процедур `read` и `readln` в ходе работы программы. Вид окна редактора представлен на рис _____

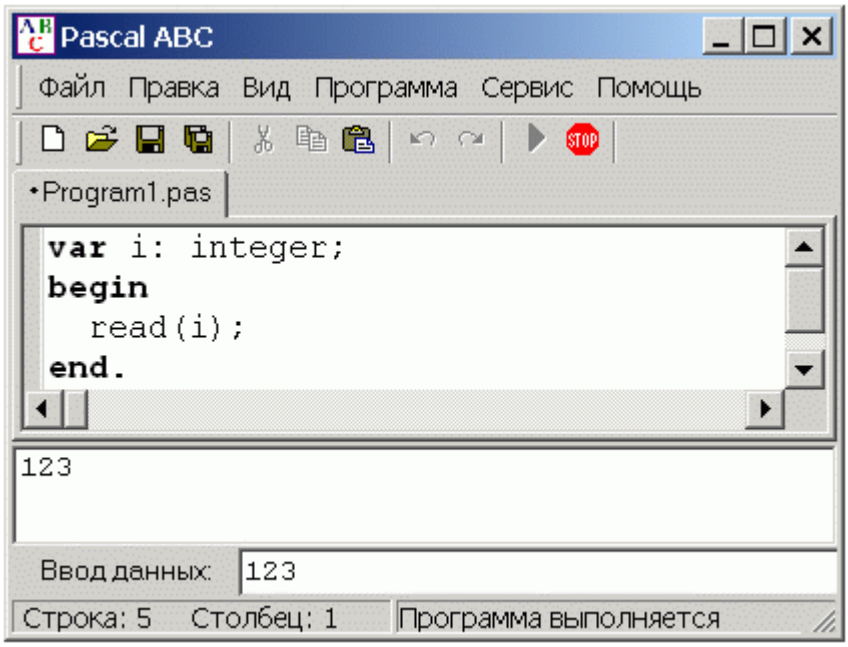


рис _____ Вид окна редактора PascalABC

Выполнение программы

Для выполнения программы в текущем окне редактора следует нажать клавишу **F9** или кнопку на панели инструментов **Выполнить программу**.

Программа вначале компилируется во внутреннее представление, после чего, если не найдены ошибки, программа начинает выполняться. При выполнении программы кнопка запуска программы становится неактивной, кнопка останова программы, наоборот, активной и в строке статуса отображается информация "Программа выполняется".

При выводе в графическое окно модуля GraphABC программу можно прервать нажатием клавиши **Esc**, при этом графическое окно будет закрыто.

Пошаговое выполнение программы

Режим пошагового выполнения предназначен для отладки программы. Для выполнения одного шага (одной строки) программы следует нажать клавишу **F8** или кнопку (шаг без входа в подпрограмму), либо клавишу **F7** или кнопку (шаг со входом в подпрограмму). Для выполнения программы до данной строки следует установить на нее курсор и нажать клавишу **F4** или кнопку .

Выполнение программы можно в любой момент прервать нажатием комбинации клавиш **Ctrl-F2** или кнопки **Завершить работу программы**. При этом в окне вывода появится сообщение

Программа прервана пользователем

Прервать программу, находящуюся в режиме пошагового выполнения, можно с помощью комбинации клавиш **Ctrl-F2** или кнопки **Завершить работу программы**. Если программа находится в режиме пошагового выполнения, то ее можно выполнить до конца, нажав **F9**.

Просмотр результатов выполнения программы осуществляется в окне **Вывода** (Для **в**ывода окна нажать клавишу **F5)**.

Интегрированная среда программирования PascalABC имеет встроенную справочную системы, которая позволяет программисту получить справочную информацию по интересующему вопросу.

Ввод данных в в окно ввода сопровождается эхо-выводом в окно вывода (см. рис). После нажатия клавиши **Enter** данные из окна ввода попадают в соответствующие переменные, окно ввода закрывается, и программа продолжает работать дальше.

Контрольные вопросы и задания

1. Что включают в себя имена данных?
2. Сколько в следующем списке зарезервированных слов:
3. X, Program, Y, Summa, MyMoney, Произведение, Vova, begin, end, if, repeat, Read?
4. Из каких разделов состоит программа?
5. Какие действия производятся при выполнении раздела VAR?
6. В каких случаях надо использовать переменные:
   a. если в программе используется какое-то число?
   b. если в вычислениях какой-то операнд постоянно меняет свое значение?
   c. если операнд в выражении хотя бы один раз меняет значение?
7. Какие заголовки программ правильны:
8. program Zarplata?
9. program Сумма?
10. program Summa Nalogov?
11. программа Teach_Kurs?
12. program 12Kurs2?
13. program Summa_Elementov?
14. Какая структура программы правильная:

1)
```
program MyProgram;
begin
   Writeln ('Привет');
     end.
```
2)
```
program MyFirst;
begin
   X:=Y+100;
   end.
```

2. Какой из перечисленных разделов обязателен в программе:
   1) раздел var?
   2) раздел const?
   3) раздел type?
   4) раздел begin .. end.?
   5) раздел label?
3. Какие из комментариев неправильны:
   1) { Программа вычисляет логарифм введенного числа};
   2) (* Это тоже комментарий *);
   3) {{ Комментарий в комментарии }};
   4) (* { И это комментарий в комментарии } *).

4. *Операторы языка* Pascal ABC

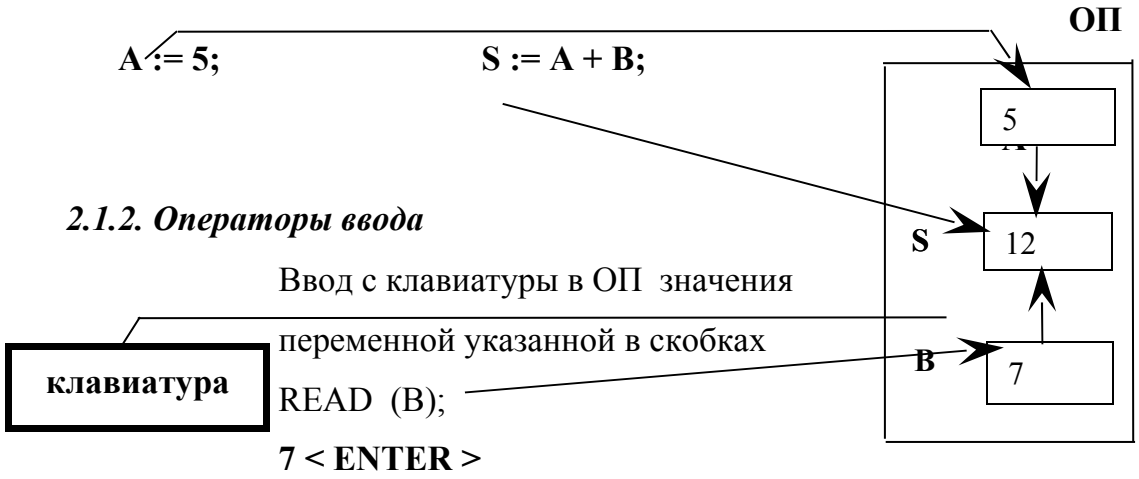
Операторы ввода-вывода

Оператор присваивания

Оператор присваивания имеет вид: переменная:= выражение

Например**:** **S := A + B;**

‘**S**’ **–** имя переменной**,** ‘**:=**’ - знак присваивания, ‘**A+B**’ **–** выражение. После выполнения такой строки в памяти ЭВМ будет записано значение или вычисленное выражение указанное после знака присваивания.



2.1.2. Операторы ввода

Ввод с клавиатуры в ОП значения переменной указанной в скобках

READ (B);

7 < ENTER >

После набора на клавиатуре цифры 7 и нажатия клавиши Enter, значение переменной указанной в скобках будет занесено в ОП (более одного значения вводится через пробел или после каждого операнда нажимают клавишу Enter).

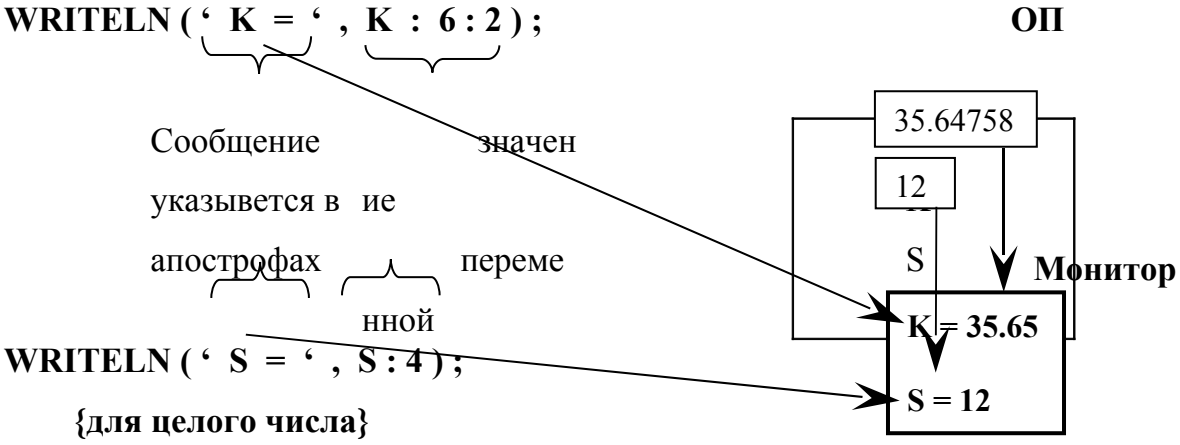
Оператор вывода

Пусть в ОП будут записаны значения переменных в ячейках:

S : = 12; - целое число

K: = 35.64758; - вещественное число

Вывод из ОП на экран сообщения и (или) значение переменной.

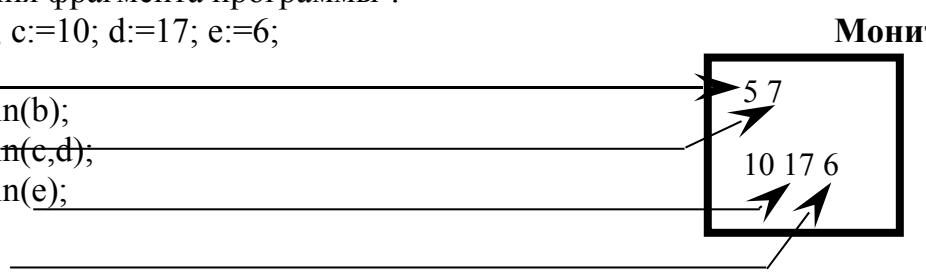


Примечание:

После выполнения операторов Read или Write указатель остается на месте вывода (ввода) данных, а после выполнения операторов Readln или Writeln

указатель перемещается на новую строку. Пример: что будет выведено на экран после выполнения фрагмента программы ?

```
A:=5; b:=7; c:=10; d:=17; e:=6;
Write(a);
    Writeln(b);
    Writeln(c,d);
    Writeln(e);
```



Форматированный вывод

Для целого числа **после имени переменной через двоеточие указывается количество позиций отводимых для вывода числа, например: WRITE ('S = ',S : 4).**

Для вещественного числа **после имени переменной через двоеточие указывается количество позиций отводимых для вывода числа, вторая цифра указывает на количество позиций после запятой, например: WRITE (' K = ' , K : 6 : 2) ;**

Контрольные вопросы и задания

Вопросы:

1. Какие процедуры служат в Паскале для выполнения операций ввода-вывода?
2. Напишите оператор ввода переменной К с клавиатуры.
3. Для каких целей служит оператор присваивания.
4. Чем отличаются операторы ввода Read и Readln?
5. Для каких целей служит оператор Write.
6. Чем отличаются операторы вывода Write и Writeln?
7. Для чего в процедурах вывода определяется ширина поля вывода?
8. Какие обозначения используются в форматах вывода?

Задания:

1. Составить программу для вычисления высот треугольника со сторонами a, b, c по формулам:

$h_{a=}\frac{2}{a}\sqrt{p(p-a)(p-b(p-c)}$; $h_{b=}\frac{2}{b}\sqrt{p(p-a)(p-b(p-c)}$; $h_{c=}\frac{2}{c}\sqrt{p(p-a)(p-b(p-c)}$

где p=(a+b+c)/2.

2. Составьте программу вычисления площади прямоугольника по введенным в диалоге двум сторонам. Запишите текст программы на диск под именем okr.pas, откомпилируйте и проверьте ее действие.

3. Составьте программу вычисления длин высот треугольника, у которого длины сторон A,B,C.

4. Составьте программу вычисления величины силы тока на участке электрической цепи сопротивлением R Ом при напряжении U В.

5. Составьте программу вычисления напряжения на каждом из последовательно соединенных участков электрической цепи сопротивлением R1, R2, R3 Ом, если сила тока при напряжении U В составляет I А.

6. Напишите программу, которая вводит значения трех переменных: A, B, C типа Real и выводит их сумму. Ввод каждого значения произвести с отдельной строки. Результат также помещается на отдельную строку. При составлении программы обеспечьте приглашение к вводу данных.

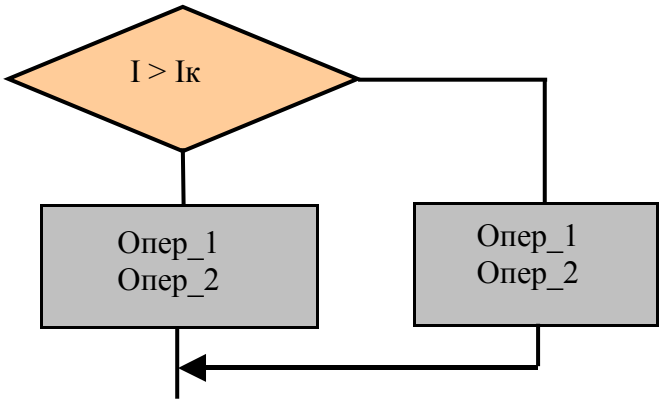
1. Составьте программу, которая выводит на экран компьютера заставку, аналогичную следующей:

****** Программа вычисления суммы чисел*
* Автор: Петров В. И *
****

8. Напишите программу, которая вводит значения четырех переменных A, B, C, D типа integer и выводит их сумму. Ввод пары значений A и B произвести на одной строке, C и D — на другой. Результат вывести на отдельную строку, и курсор оставить на той же строке.

Программирование разветвляющихся процессов

Ветвление полное:



Условный оператор IF используется для изменения естественного порядка выполнения операторов программы. Если условие истина, то выполняется первая ветвь, иначе – вторая. Таким образом, условный оператор – это средство ветвления вычислительного процесса.

Составной оператор IF имеет 2 формы: полное ветвление и сокращенное. В сокращенном ветвлении вторая ветвь отсутствует.

Примечание:

1. В операторе IF перед ELSE точка с запятой не ставится.
2. Условный оператор управляет только одним оператором поэтому, если после ключевых слов Then и Else требуется произвести более одного действия, то необходимо использовать операторные скобки Begin End.
3. Внутри операторных скобок после каждого оператора точка с запятой ставится.

Пример выполнения задачи на полное ветвление

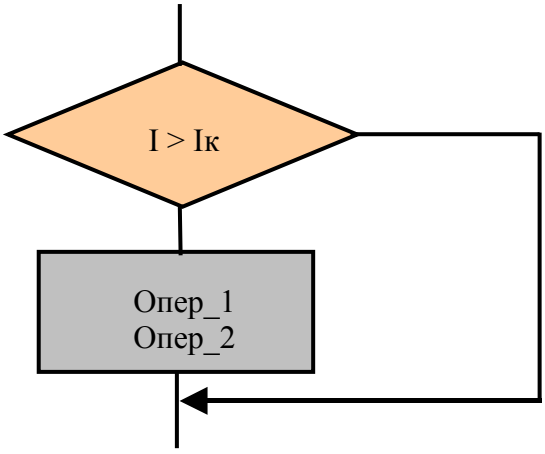
Задача №2. Вычислить корни квадратного уравнения общего вида $ax^2 + bx + c = 0$ в области действительных чисел.

Программа имеет вид :

```
(* ОПРЕДЕЛЕНИЕ КОРНЕЙ КВАДРАТНОГО УРАВНЕНИЯ *)
PROGRAM KU;                                  {Имя программы}
 VAR                                         {Раздел описания переменных}
       A,B,C:INTEGER;                        {коэффициенты уравнения}
       D,X1,X2:REAL;                         {Дискриминант и корни уравнения}
BEGIN
 WRITE('ВВЕДИТЕ КОЭФФИЦ. A,B,C  '); {Вывод сообщения}
 READ (A,B,C);                               {Ввод данных с клавиатуры}
 WRITELN ('A=',A,'B=',B,'C=',C);             {Эхо-печать ввода исходных данных}
 D:=SQR(B)-4*A*C;                            {Вычисление дискриминанта}
 IF D>0 THEN                                 {Проверка выполнения условия}
     BEGIN
        X1:=(-B+SQRT(D))/(2*A);              {Выполняемые действия }
        X2:=(-B-SQRT(D))/(2*A);              {если условие ИСТИНА}
        WRITELN ('X1=',X1,'X2=',X2);         {Вывод результата }
      END
        ELSE IF D=0 THEN              {ИНАЧЕ, Проверка выполнения условия}
           BEGIN
              X1:=(-B+SQRT(D))/(2*A);              {Выполняемые действия }
              X2:=X1;                              {если условие ИСТИНА}
              WRITELN ('X1=',X1,'X2=',X2);
           END
              ELSE  WRITELN ('НЕТ РЕШЕНИЯ');  {если условие ложь}
END.                                           {Конец программы}
```

Если вторая ветвь отсутствует, тогда имеет место сокращенное ветвление. Фрагмент программы представлен ниже.

2.2.2. Ветвление сокращенное:

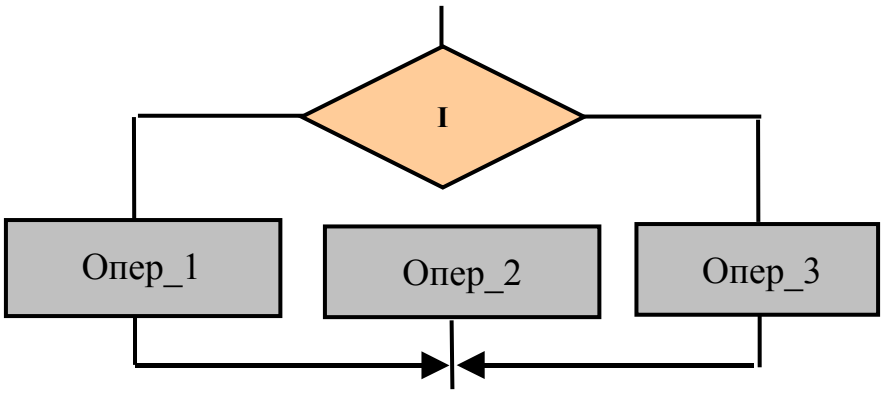



```
IF I < = Iк  THEN
   BEGIN
      Опер_1;
      Опер_2;
    END;
```

2.2.3. Оператор выбора:

Оператор **case** работает следующим образом. Сначала

Оператор **case** работает следующим образом. Если в одном из списков выбора найдено текущее значение переключателя, то выполняется оператор, соответствующий данному списку. Если же значение переключателя не найдено ни в одном списке, то выполняется оператор по ветке **else** или, если ветка **else** отсутствует, оператор **case** не выполняет никаких действий, далее выполняется оператор стоящий за словом end.




```
CASE  I  OF
        1:  Опер_1;
        2:  Опер_2;
        3:  Опер_3;
    ELSE Опер 4;
END;
```

Пример выполнения задачи на использование оператора выбора

Задача №3. Составить

Список выбора состоит либо из одной константы, либо из диапазона значений вида **a..b** (константа **a** должна быть меньше константы **b**); можно также перечислить несколько констант или диапазонов через запятую:

Пример выполнения задачи на использование оператора выбора

Задача №3. Составить программу для ввода на экран монитора номера дня недели и вывода соответствующего ему дня недели на русском языке.

Программа решения задачи имеет вид:

```
PROGRAM DNED;                                  {заголовок программы}
    VAR                                        {раздел описания переменных}
      N:INTEGER;
    BEGIN
      WRITELN ('ВЫВЕДИТЕ НОМЕР ДНЯ НЕДЕЛИ');{Вывод сообщения}
      READ(N);                                 {Ввод значения n с клавиатуры}
      CASE N OF                                {Выбор  варианта }
        1:WRITELN('понедельник');              { Выполняемые операторы  }
        2:WRITELN('вторник');          {в зависимости от значения селектора}
        3:WRITELN('среда');
        4:WRITELN('четверг');
        5:WRITELN('пятница');
        6:WRITELN('суббота');
        7:WRITELN('воскресенье');
      END;                                     { Конец оператора  Case}
  END.                                         {Конец программы}
```

Контрольные вопросы и задания

Вопросы:

1. Что представляет собой составной оператор? Как ограничиваются операторы, объединенные в составной оператор?
2. Назначение, формы записи и порядок выполнения оператора условия if.
3. Особенности использования вложенных условных операторов.
4. Каковы отличия оператора выбора case от оператора условия if?
5. Для чего служит ключ выбора и какого он может быть типа.
6. Сколько меток может быть перед оператором в списке выбора.

Задания:

1. Составьте программу, реализующую эпизод применения компьютера в книжном магазине. Компьютер запрашивает стоимость книг, сумму денег, внесенную покупателем; если сдачи не требуется, печатает на экране "спасибо"; если денег внесено больше, то печатает "возьмите сдачу" и указывает сумму сдачи; если денег недостаточно, то печатает об этом сообщение и указывает размер недостающей суммы.

2. В ЭВМ поступают результаты соревнований по плаванию для трех спортсменов. Составьте программу, которая выбирает лучший результат и выводит его на экран с сообщением, что это результат победителя заплыва.

3. Ввести два числа. Меньшее заменить полусуммой, а большее - удвоенным произведением.

4. Вычислить

$$y = \begin{cases} \sin X, & \text{при } X>0 \\ \operatorname{tg} X, & \text{при } X<=0 \end{cases}$$

5. Составить программу для вычисления значений функции:

$$y = \begin{cases} x - 1 & \text{при } x \le 2; \\ x^2 & \text{при } 2 < x < 3; \\ 6 & \text{при } 3 \le x \le 5. \end{cases}$$

2.3. Программирование циклов

Для всех операторов цикла характерно следующая особенность. Повторяющиеся вычисления записываются всего лишь один раз. Вход в цикл возможен только через его начало. Переменные оператора цикла должны быть определены до входа в циклическую часть. Необходимо предусмотреть выход из цикла: или по естественному его окончанию, или по оператору перехода.

Цикл содержит:

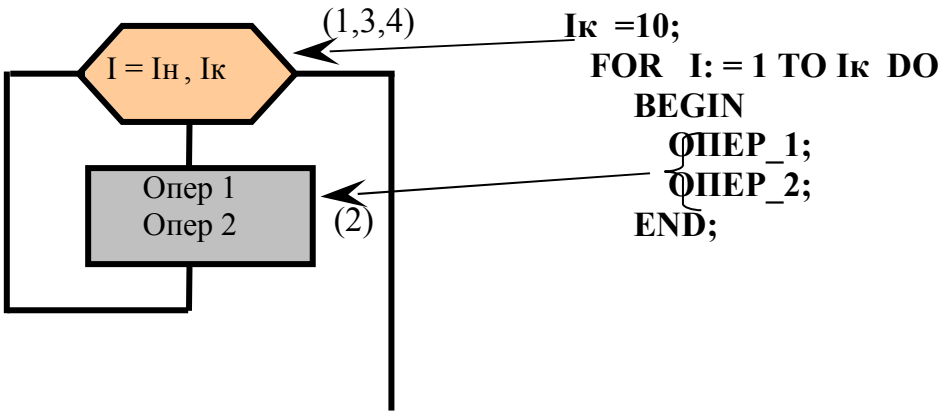
1. подготовку (начало) цикла. (Управляющую переменную, ее начальное, конечное значение и шаг приращения);
2. тело цикла (повторяющиеся операторы);
3. изменение значения управляющей переменной на величину шага;
4. проверку на окончание цикла.

2.3.1. Цикл с параметром

Оператор цикла с параметром используется в тех случаях, когда заранее известно, сколько раз должна повторяться циклическая чагсть программы. Оператор цикла имеет вид:

Оператор цикла **for** имеет одну из двух форм:

for переменная:=начальное значение **to** конечное значение **do**
оператор
или
for переменная:=начальное значение **downto** конечное значение **do**
оператор



При программировании циклов с параметром необходимо помнить следующие ***правила организации цикла***:

1) параметр цикла, начальное и конечное значения, должны быть одинакового типа, их тип может быть любым скалярным типом (стандартным, перечисляемым, ограниченным), кроме вещественного;

2) очередное значение параметра вычисляется автоматически, для целого типа шаг изменения значения параметра цикла равен 1 при TO и -1 при DOWNTO;

3) запрещено изменять внутри тела цикла значение управляющей переменной цикла;

4) запрещено входить в цикл с помощью оператора GOTO, минуя оператор FOR,;

5) цикл не выполняется вообще, если начальное значение больше (при DOWNTO - меньше), чем конечное;

6) после служебного слова DO может стоять только один оператор; если в цикле нужно выполнить группу операторов, то их заключают в скобки BEGIN-END;

7) из составного оператора, входящего в оператор цикла, можно выйти до окончания этого цикла с помощью оператора GOTO, тогда последнее значение параметра цикла сохраняется.

Пример использования цикла с параметром

Задача № 4. Вычислите степень y = a действительного числа a с натуральным показателем n. Воспользуемся для вычислений следующей формулой:

До начала цикла (подготовка цикла) y:= 1, на каждом шаге цикла (для i = 1,2,...,n) - y:= y*a. Цикл с параметром i будет выполнен n раз.

Программа решения задачи имеет вид:

```
(*ОПРЕДЕЛЕНИЕ СТЕПЕНИ ВЕЩЕСТВЕННОГО ЧИСЛА*)
PROGRAM STEPEN;              {Имя программы}
VAR                            {Раздел описания переменных}
A,Y:REAL;
I,N:INTEGER;
BEGIN (*STEPEN*)
WRITELN ('ВВЕДИТЕ ЧИСЛО И СТЕПЕНЬ ЧИСЛА'); {Вывод  оообщения}
READ (A,N);                                  {Ввод значения
переменных}
Y:=1;                                        {Ввод начального значения}
FOR I:=1 TO N DO                           { цикл для вычисления степени}
```

```
Y:=Y*A;                                    {Вывод сообщения}
WRITELN (N,'СТЕПЕНЬ ЧИСЛА',A);          {Вывод сообщения}
WRITELN ('РАВНА',Y)                       {Вывод результата}
END.(*STEPEN*).                            {Конец программы}
```

2.3.2. Цикл с предусловием

Оператор цикла **с предусловием** имеет следующую форму:

```
while условие do
оператор
```

Условие представляет собой выражение логического типа, а оператор после **do** называется *телом цикла*.

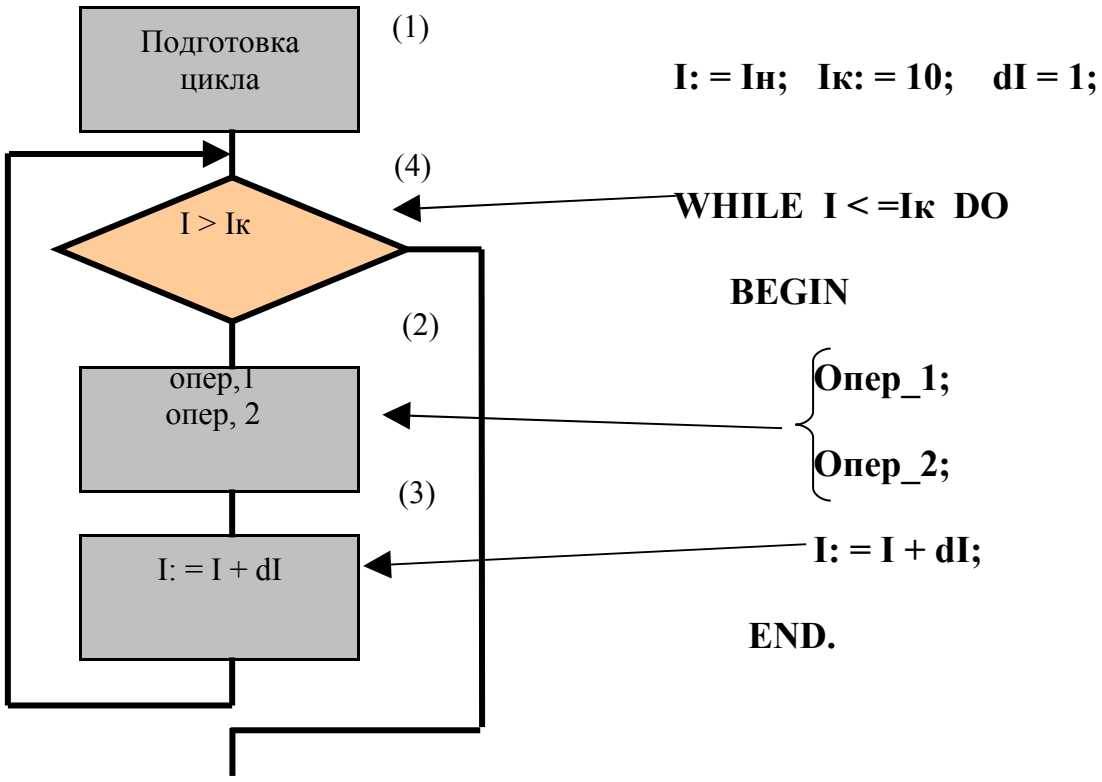
Перед каждой итерацией цикла условие вычисляется, и если оно истинно, то выполняется тело цикла, в противном случае происходит выход из цикла.

Цикл с предусловием используется, как правило, в тех случаях, когда заранее неизвестно число повторений цикла. В теле цикла с предусловием и постусловием необходимо указывать изменение управляющей переменной на величину шага. Чтобы прервать зациклившуюся программу, следует использовать комбинацию клавиш **Ctrl-F2** или кнопку **STOP.** Если в теле цикла операторов более одного, то тело цикла заключается в операторные скобки

Форма записи оператора цикла с предусловием:

Здесь WHILE (Пока) DO (выполнить) – служебные слова.

До начала циклов с предусловием и постусловием необходимо указывать начальное значение управляющей переменной.



Пример использования цикла с предусловием

Задача № 5. Составить программу для вычисления значения функции $y = ax^2$. Переменная x изменяется от 5 до 25 с шагом 1, полученный результат вывести на экран.

Программа решения задачи имеет вид:

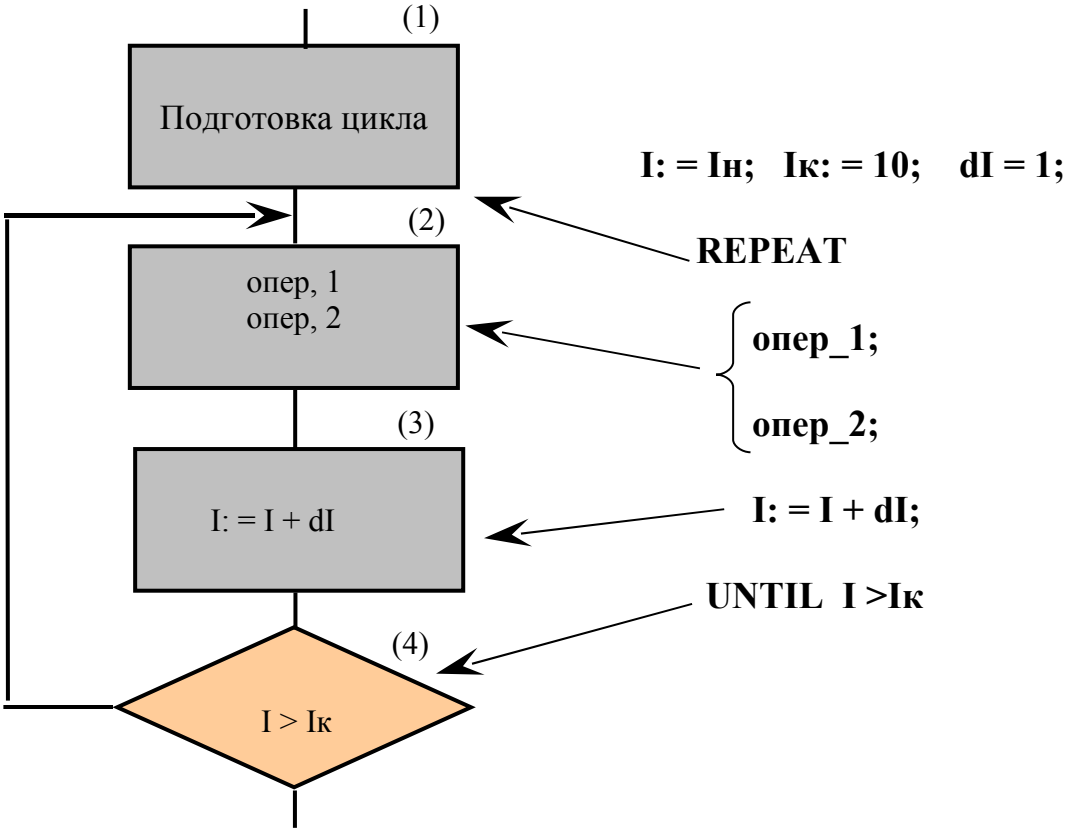
```
program pred5;                              {заголовок программы}
 var                                        {раздел описания переменных}
   x:integer;
   a,y:real;
 begin
    writeln ('Введи параметр a ');          {Вывод комментария}
    readln (a);                             {Ввод с клавиатуры значения a}
    y:=0;                       {Обнуление переменной для накапливания суммы}
    x:=5;                                   {Задание начальных условий}
      while x<=25 do                        {Начало  цикла с предусловием}
        begin;
           y:=a*x*x;                        {Тело цикла с предусловием}
           x:=x+1;              {изменение переменной цикла на величину шага}
        end;                                {конец цикла}
    writeln ('Функция равна ',y);           {вывод результата}
 end.                                       {Конец программы}
```

Задача № 5. Составить программу для вычисления значения функции $y = ax^2$. Переменная x изменяется от 5 до 25 с шагом 1, полученный результат вывести на экран.

Программа решения задачи имеет вид:

```
program pred5;                              {заголовок программы}
 var                                        {раздел описания переменных}
   x:integer;
   a,y:real;
 begin
    writeln ('Введи параметр a ');          {Вывод комментария}
    readln (a);                             {Ввод с клавиатуры значения a}
    y:=0;                       {Обнуление переменной для накапливания суммы}
    x:=5;                                   {Задание начальных условий}
      while x<=25 do                        {Начало  цикла с предусловием}
        begin;
           y:=a*x*x;                        {Тело цикла с предусловием}
           x:=x+1;              {изменение переменной цикла на величину шага}
        end;                                {конец цикла}
    writeln ('Функция равна ',y);           {вывод результата}
 end.                                       {Конец программы}
```

2.3.3. Цикл с постусловием



Цикл с постусловием используется, как правило, в тех случаях, когда заранее неизвестно число повторений цикла.

Пример использования цикла с постусловием

Задача 6. Составить программу для вычисления значения функции $y = ax^2$. Переменная x изменяется от 5 до 25 с шагом 1, полученный результат вывести на экран.

Программа решения задачи имеет вид:

```
program post3;                          {заголовок программы}
  var                                   {раздел описания переменных}
    x:integer;
    a,y:real;
 begin
    writeln ('Вводи параметр a ');      {Вывод сообщения}
    readln (a);                         {Ввод с клавиатуры значения a}
    y:=0;                               {Обнуление переменной для накапливания
суммы}
    x:=5;                               {Задание начальных условий}
     repeat;                            {Начало цикла с постусловием}
       y:=a*x*x;                        {Тело цикла с постусловием}
       x:=x+1;                          {изменение переменной цикла на величину шага}
     until x>25;                        {проверка условия на выход из цикла}
  writeln ('Функция равна ',y);         {вывод результата}
end.                                    {Конец программы}
```

Контрольные вопросы и задания

Вопросы:

1. Что такое цикл, управляющая переменная цикла?
2. Оператор цикла с параметром.
3. Оператор цикла с предусловием.
4. Оператор цикла с постусловием.
5. Отличия цикла с параметром от других операторов цикла.
6. Отличия цикла с постусловием от других операторов цикла.

Задания:

1. Составить программу для вычисления суммы ряда:

$$\sum_{n=1}^{45} \frac{1}{n}$$

2. Составить логическую схему алгоритма и для вычисления значения функции: y=2*x*x; при переменной x изменяющейся от 1 до 3 с шагом 0.1.

3. Составьте программу, которая вычисляет сумму чисел от 1 до N. Значение N (N должно быть меньше 100) вводится с клавиатуры.

4. Напишите программу печати таблицы перевода расстояний из дюймов в сантиметры (1 дюйм = 2,5 см) для значений длин от 1 до 20 дюймов.

5. С помощью while напишите программу вывода всех четных чисел в диапазоне от 2 до 100 включительно.

6. Составьте и отладьте программу, вычисляющую сумму квадратов чисел от 1 до введенного вами целого числа n.

7. С помощью while напишите программу определения суммы всех нечетных чисел в диапазоне от 1 до 99 включительно.

8. С помощью цикла while напишите программу определения идеального веса для взрослых людей по формуле: Ид.вес = рост—100. Выход из цикла: значение роста = 250.

9. С помощью repeat напишите программу-фильтр, которая вводит любые символы, но комментирует только буквы русского алфавита. Завершение работы программы — по нажатии буквы "Я".

10. С помощью repeat напишите программу, которая требует у вас пароль, например 111, и если пароль правильный, то заполняет все строки экрана сообщением "Молодец!!!". Если после пятой попытки пароль все равно неверен, выйти из программы.

11. Составьте и отладьте программу, определяющую максимальное из всех введенных вами чисел. (Пусть признаком конца ввода чисел является введенное число 0.)

2.4. Массивы

При использовании больших объемов данных требуется как-то их структурировать или объединить данные в отдельные группы. Решить такую задачу можно путем использования массивов. **Массив** – это упорядоченная совокупность значений одинакового типа. Например, в программе можно описать и обрабатывать массивы целых чисел, логических и символьных значений. Массивы могут быть одномерными, двумерными и многомерными.

2.4.1. Одномерный массив

Описание массива

1. VAR < имя >; ARRAY [тип индекса] OF < тип элемента >

Пример:

VAR

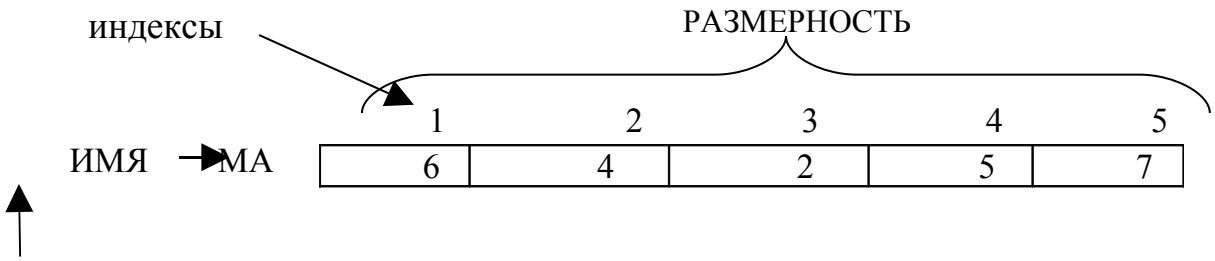
MA: ARRAY [1..5] OF INTEGER;

2. TYPE < имя типа> = ARRAY [1..5] OF < тип элемента >

VAR < имя массива > < имя типа>

Пример:

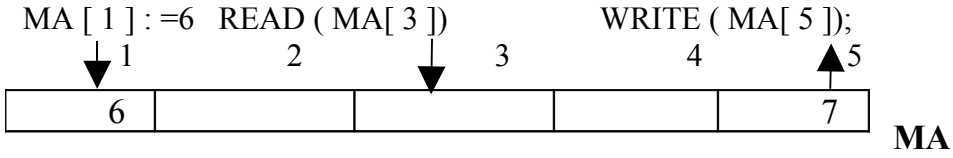
```
TYPE M = ARRAY [1..5] OF INTEGER;
VAR MA:M;
```



ТИП ЭЛЕМЕНТА И ЕГО ЗНАЧЕНИЕ

ВВОД ЗНАЧЕНИЙ ЭЛЕМЕНТА МАССИВА

ВЫВОД ЗНАЧЕНИЙ ЭЛЕМЕНТА МАССИВА



2.4.2. Двумерный массив

1. VAR < имя >; ARRAY [тип индекса строк, тип индекса столбцов] < тип элемента >

Пример:

VAR

MA2: ARRAY [1..3,1..5] OF REAL;

2. TYPE M = ARRAY [1..3,1..5] OF REAL;

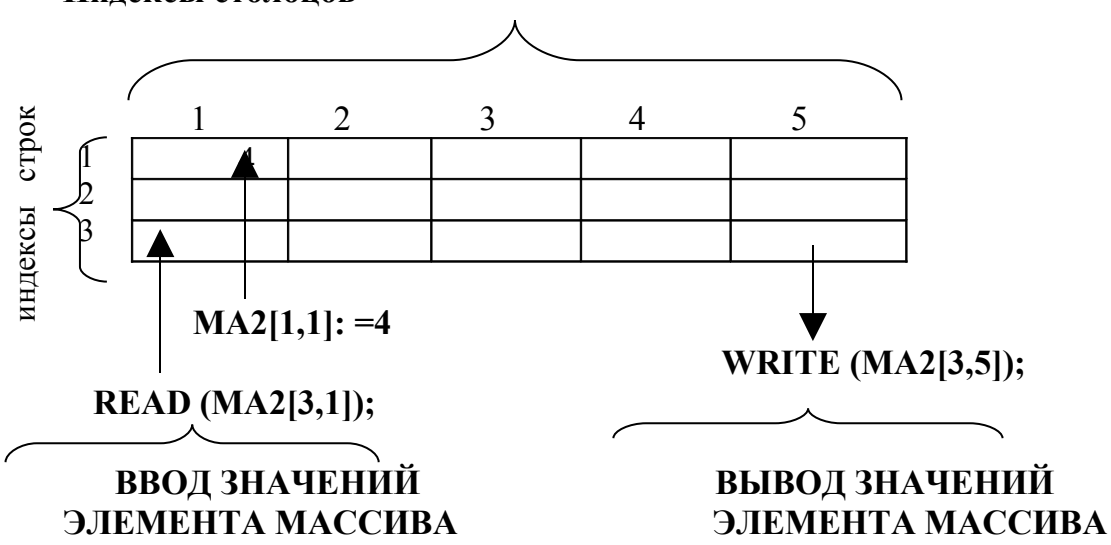
VAR MA2:M:

. . .

BEGIN

. . .

```
. . .  READ (MA2[1,3]);
       WRITE  (MA2[3,5]);
. . .          End.
```



Пример использования одномерного массива

Задача № 7. Составить программу вычисления суммы действительных чисел, записанных в одномерный массив a, размером 7 элементов и вывода на экран содержимого введенного массива и полученной суммы.

Программа решения задачи имеет вид:

```
program massiv;                          {заголовок программы}
     const x=7;                          {описание константы}
 var                                     {Раздел описания переменных}
    s:real;
    i:integer;
    a:array[1..x] of real;               {описание массива}
begin
   writeln('Введите ',x,' чисел');       {Вывод сообщения}
          for i:=1 to x do               {Цикл для ввода элементов массива}
           readln(a[i]);                 {ввод с клавиатуры значений в массив}
   s:=0;                        {Обнуление переменной для накапливания суммы}
        writeln('      Введенный массив ');
    for i:=1 to x do                {Начало цикла для вывода элементов массива}
           begin
              write(a[i]:5:1,' ');      {вывод значений массива на экран}
              s:=s+a[i];                {Накапливание суммы элементов массива}
           end;                         {Конец цикла}
     writeln;                           {переход на новую строку}
     writeln('Сумма элементов массива s=  ',s:5:1);
end.                                      {Конец программы}
```

Пример задачи с двумерным массивом

Задача № 8. Набрать, отредактировать, отладить и выполнить программу формирования единичной матрицы M2(10*10).

Программа решения задачи имеет вид:

```
program mas_2;                              {заголовок программы}
var
   i,j:integer;
  M2:array[1..10,1..10] of integer;         {описание массива}
begin
      for i:=1 to 10 do              {Цикл для ввода элементов массива по строкам}
        for j:=1 to 10 do            {Цикл для ввода элементов массива в строке}
          if i=j Then M2[i,j]:=1 Else M2[i,j]:=0; {ввод  значений элементов
массива}
       writeln ('      Единичный массив ');
      for i:=1 to 10 do            {Цикл для вывода элементов массива по строкам }
begin                               {начало цикла по строкам }
            for j:=1 to 10 do  {Цикл для вывода элементов массива по
элементам строки}
              write(M2[i,j]:5,' ');     {вывод значений массива строки на экран}
Writeln;                                {переход на новую строку }
End;                                    {конец цикла по строкам }
end.                                    {Конец программы }
```

Контрольные вопросы и задания

Вопросы:

1. Что такое массив?
2. Как определить местоположение элемента в массиве?
3. Что такое индекс? Каким требованиям он должен удовлетворять?
4. Особенности расположения элементов массива в памяти ЭВМ.
5. Каким образом задается описание массива, что в нем указывается?
6. Общие и отличительные черты одномерных, двумерных и n-мерных массивов.
7. В каких операциях могут участвовать массивы и какие к ним при этом предъявляются требования?
8. Каким образом в Паскале задается обращение к элементу массива?

Задания:

1. Введите с клавиатуры в массив X пять целочисленных значений, выведите их в одну строку через запятую; получите для массива среднюю арифметическую.
2. Введите с клавиатуры пять целочисленных элементов массива X. Выведите на экран значения корней и квадратов каждого из элементов массива.
3. Создайте массив из пяти фамилий и выведите их на экран столбиком, начиная с последней.
4. Создайте массив из пяти фамилий и выведите на экран те из них, которые начинаются с определенной буквы, которая вводится с клавиатуры.
5. Дан одномерный массив. Вставьте в него элемент L в позицию K.

6. Введите с клавиатуры целочисленные элементы матрицы 3x3, выведите исходную матрицу на экран. Умножьте каждый элемент матрицы на 3 и выведите результат на экран.

7. Создайте двумерный массив (3x4) целых чисел и найдите сумму всех его элементов.

8. Введите с клавиатуры целочисленные элементы матрицы 3x3 и вычислите сумму элементов каждого столбца.

10. Создайте массив из 15 целочисленных элементов и определите среди них минимальное значение.

11. Создайте двумерный массив X, имеющий четыре строки и три столбца и найдите в нем максимальный по абсолютному значению элемент, а также укажите номер строки и столбца, содержащие этот элемент. Например, в массиве

```
 2      1   3
-4   0  8
 7      5   1
-3   1  0
```

максимальный по абсолютному значению элемент = 8, находится он во второй строке третьего столбца.

12. Введите массив (не более 20) и определите, есть ли в нем элементы с одинаковыми значениями.

2.5. Подпрограммы

При разработке программ иногда требуется одни и те же последовательности действий выполнять на различных этапах обработки информации. В таких задачах в различных местах встречаются фрагменты, одинаковые по выполняемым действиями, различающихся только в значениях исходных данных. Повторяющаяся группа операторов оформляется в виде самостоятельной программной единицы – подпрограммы. Подпрограмма - это самостоятельная часть программы, реализующая определенный алгоритм и допускающая обращение к ней из различных частей основной программы.

В языке Паскаль подпрограммы реализуются в виде процедур и функций, которые вводятся в программу с помощью своего описания.

2.5.1. Процедуры

Процедуры описываются в специальном разделе описательной части программы вслед за разделом переменных. Любая процедура состоит, аналогично программе, из заголовка процедуры и тела процедуры.

Заголовок процедуры представляет собой:

PROCEDURE < и м я> (список параметров);

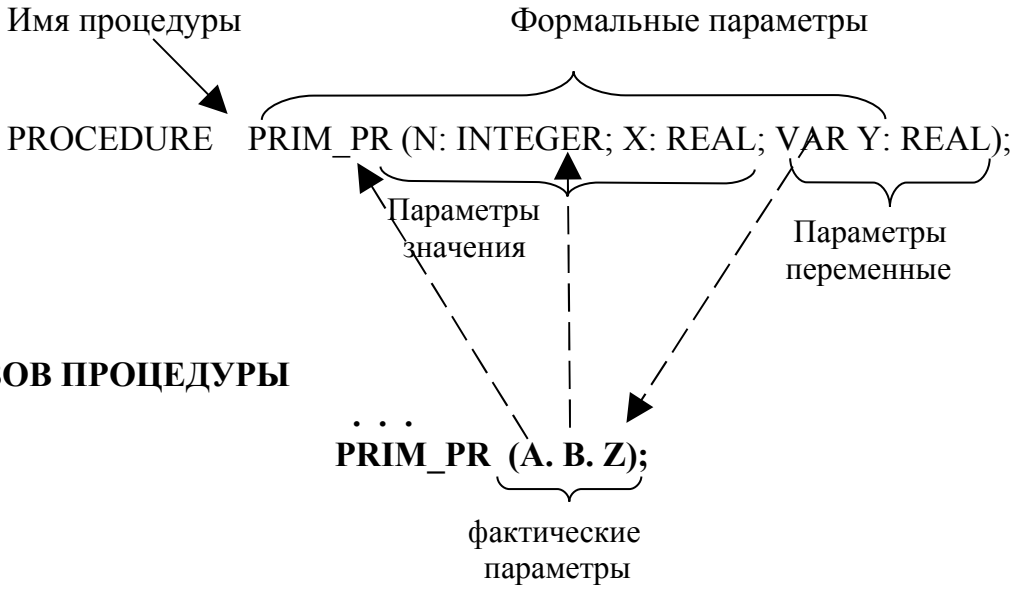
где PROCEDURE – служебное слово;

имя – имя процедуры, определяемое в соответствии с правилами построения идентификаторов;

список параметров - перечень имен для обозначения исходных данных и результатов работы процедуры с указанием их типов.

Пример:

1. ЗАГОЛОВОК ПРОЦЕДУРЫ.



2.ВЫЗОВ ПРОЦЕДУРЫ

```
. . .
PRIM_PR (A. B. Z);
```

фактические параметры

Рассмотрим задачу с использованием процедуры

Задача 9. Составить программу вычисления степени $Z=a^m$, где m любое целое (положительное или отрицательное) число и ‘a’ не равно нулю. Для решения использовать процедуру с параметром.

$$Z = \begin{cases} a^m, & \text{если } m>0; \\ 1, & \text{если } m=0; \\ 1/(a)^m, & \text{если } m<0. \end{cases}$$

учитывая, что $1/(a)^{-m} = (1/a)^{-m}$ и используя процедуру с параметром составим программу:

Программа решения задачи имеет вид:

```
program step;                                  {заголовок программы}
   var                                         {описание переменных}
      m:integer;                               {показатель степени}
      a,z:real;                                { число,    результат }
procedure step1 (n:integer; x:real;var y:real);   {заголовок процедуры}
      var
       i:integer;
  begin                                        {операторная часть процедуры}
   y:=1;
      for i:=1 to n do                         {цикл для вычисление степени}
         y:=y*x;                               {цикл для вычисление степени}
  end;                                         {конец процедуры}
begin
  writeln (' Введите  a,m ');                  {вывод сообщения}
  readln (a,m);                                {ввод с клавиатуры значений в массив}
    If m=0 then z:=1                          {проверка условия, выполнение оператора}
      else if m>0 then step1(m,a,z)  {иначе проверка условия, выполнение
оператора}
              else step1(-m,1/a,z);          {иначе выполнение оператора}
  writeln (a:4:2,' в степени ',m:3,' равно ',z:4:2);  {вывод результата}
  end.                                         {конец программы}
```

2.5.2. Функции

Функция – это подпрограмма, результат выполнения которой есть единственное скалярное значение, присваиваемое имени этой функции. Функция является частным случаем процедур. Отличия процедуры от функции:

- результат выполнения функции – одно значение, а процедуры одно или несколько;
- результат выполнения функции передается в основную программу как значение имени этой функции, а результаты выполнения процедуры – как значение ее параметров.

Заголовок функции представляет собой:

FUNCTION < и м я> : тип;

где FUNCTION – служебное слово;

имя – имя функции;

тип - тип результата значения, которое должно приобретать имя функции

ПРИМЕР

1. ЗАГОЛОВОК ФУНКЦИИ.

FUNCTION F (N: REAL): REAL;

2. ВЫЗОВ ФУНКЦИИ.

PER: = F (K);

Пример решения задачи с использованием функции

Задача № 10. Составьте программу вычисления факториалов Fn=n!, Fm=m!, Fnm=(n-m)!. Вычисление факториала оформить в виде функции с параметрами.

Факториал n! представляет собой произведение n чисел натурального ряда: 1*2*3*...*n.

Программа решения задачи имеет вид:

```
 PROGRAM FUNC;
      VAR
       FN,FM,FNM:INTEGER;
       N,M:INTEGER;
       (* ФУНКЦИЯ   FAKT *)
      FUNCTION FACT(K:INTEGER):INTEGER;      { начало описания функции }
        VAR
          P,I:INTEGER;          {Раздел описания  локальных переменных}
        BEGIN                   { начало операторной части функции }
          P:=1;
          FOR I:=1 TO K DO
            P:=P*I;
          FACT:=P;
        END;                    { конец описания функции }
        (* ОСНОВНАЯ ПРОГРАММА *)
 BEGIN
        WRITE('ВВЕДИТЕ ЗНАЧЕНИЯ N,M:  ');
        READ(N,M);                      {Ввод данных с клавиатуры}
        FN:=FACT(N);                    {обращение к функции }
```

```
    FM:=FACT(M);                {обращение к функции }
    FNM:=FACT(N-M);             {обращение к функции }
    WRITELN('FN=',FN:5);        {Вывод результата }
    WRITELN('FM=',FM:5);        {Вывод результата }
    WRITELN('FNM=',FNM:5);      {Вывод результата }
END.                            {Конец программы}
```

ПРИ ИСПОЛЬЗОВАНИИ ПОДПРОГРАММ БЕЗ ПАРАМЕТРОВ:

1. Глобальные переменные объявленные в основной части программы доступны во всех процедурах программы.

2. Локальные переменные объявленные в подпрограммах доступны только в данной подпрограмме и внутренних подпрограммах, но не доступны в основной части программы.

Контрольные вопросы и задания

Вопросы:

1. Что называется подпрограммой? В чем состоит сходство и различие подпрограмм-процедур и подпрограмм-функций в языке Турбо Паскаль?

2. В чем различие между стандартными и определенными пользователем подпрограммами? Приведите примеры.

3. Опишите последовательность событий при вызове процедуры или функции.

4. Что называется параметром, и каково его назначение? Формальные, фактические параметры, их взаимосвязь.

5. Каковы отличия параметров-значений от параметров-переменных, особенности их описания и применения.

6. Каковы особенности параметров-процедур и параметров-функций?

7. Чем отличаются локальные и глобальные параметры? Какова область их действия?

Задания:

1. Напишите программу вычисления выражения
   $y=(tg(X)+sin(X))*e^{cos(x)}$ при X=3.7. Результат вывести в формате 5:2.

2. Напишите программу, которая с помощью функции Chr выведет на экран кодовую таблицу ПЭВМ (ASCII-таблицу). Задержите выведенную информацию на 5 с и очистите экран.

3. Напишите программу, которая выведет на экран 10 строк по 5 случайных чисел в диапазоне 0..36.

4. Напишите программу, которая по значениям двух катетов вычисляет гипотенузу и площадь треугольника.

5. Напишите функцию возведения в степень по формуле: $A^B = Exp(Ln(A)*B)$
   и используйте ее в программе для возведения в 4-ю степень вещественного числа 2,87.

6. Оформите процедуру Proverka проверки пользователя на право работы с программой. Используйте для этого пароль = SCHOO1. Если пароль неправильный, выйти из программы по Halt.

7. Напишите программу, состоящую из трех процедур и основной программы. Первая процедура организует ввод двух целых чисел X и Y, вторая вычисляет их сумму, третья выводит результат. Используйте эти процедуры в основной программе. Используйте X,Y как глобальные переменные. Эта программа

послужит прообразом всех ваших будущих программ, т.к. в ней реализуется принцип работы любой системы: логически выделенные ввод, обработка и вывод результата.

8. Напишите программу вычисления площади поверхности и длины экватора на основе известного радиуса планет солнечной системы. Форму планет будем считать шаром. Вычисление площади и длины экватора оформите отдельными функциями.

9. Составить программу поиска большего из четырех чисел с использованием подпрограммы поиска большего из двух.

10. Даны координаты вершин многоугольника (x1,y1, x2,y2,..., x10,y10). Определить его периметр (вычисление расстояния между вершинами оформить подпрограммой).

11. Вычислить сумму: 1! + 2! + 3! +...+ n!, используя функцию вычисления факториала числа k!

12. Составьте программу вычисления числа сочетаний из N по M. Число сочетаний определяется по формуле N!/(M!*(N—M)!), где N — количество элементов перебора. Используйте подпрограмму вычисления факториала.

13. Определить НОД трех натуральных чисел.

14. Даны действительные числа s,t. Составить программу вычисления выражения

f(t,—2s,1.17) + f(2.2,t, s—t) , где f(a,b,c) = (2a—b—sin(c))/(5+|c|).

15. Составьте программу вычисления суммы квадратов простых чисел, лежащих в интервале (M,N).

Работа с графикой

Графическое изображение представляет собой совокупность отдельных точек – пикселей, которые можно закрасить в различные цвета. Каждый пиксель имеет две координаты: X и Y. Ось OX направлена слева направо, начиная с 0, а ось OY – сверху вниз, также начиная с 0. То есть левый верхний пиксель имеет координаты (0,0). В состав стандартных средств языка Паскаль не входят операторы для работы с графикой. В каждой реализации транслятора для этого предназначена специальная библиотека, содержащая процедуры и функции для работы в графическом режиме, как в системе программирования ТурбоПаскаль, или с графическим окном системы PascalABC. Для использования средств соответствующей библиотеки необходимо в начале программы выполнить ее подключение следующим образом:

Uses имя библиотеки;

В системе PascalABC библиотека графики GraphABC подключается так:

Uses GraphABC;

Описание некоторых, часто используемых, процедур и функций, входящих в библиотеку GraphABC, приведено в Приложении. Более подробную информацию можно получить в справке по PascalABC(клавиша F1) или в [2].

В системе PascalABC графическое окно выводится как дочернее окно. При этом остаются доступными средства консольного ввода–вывода информации. Для переключения между окнами следует использовать клавишу F6 или соответствующие команды меню.

Процедурный тип

Процедурный тип описывает класс процедур или функций, имеющих однотипные заголовки, т.е. однотипные списки параметров. Имена формальных параметров не являются существенными в этих описаниях. Важно только их количество, порядок следования, типы и способ передачи, а также тип результата для функции.

Определение процедурного типа аналогично заголовку подпрограммы, но при этом имя подпрограммы не задается. Например,

```
type PROC1 = procedure (a, b, c: real; var d: real);
     PROC2 = procedure;
     FUNC  = function (x: real):real;
     FUNC1 = function : real;
```

Введенный процедурный тип может быть использован при описании формальных параметров подпрограмм. Например,

```
procedure MENU (x, y: integer; SHOW: PROC2);
function      SUMMA(EPS: real; F: FUNC);
```

Если подпрограмма имеет параметр процедурного типа, то при ее вызове в качестве фактического параметра должна использоваться процедура (функция), заголовок которой соответствует заголовку, описанному в процедурном типе. Это должна быть обязательно пользовательская процедура (функция), использовать в качестве фактического параметра стандартные функции нельзя.

Например, функции с заголовками, которые соответствуют типу FUNC:

```
function F1(x: real):real;
function G2(x: real):real;
```

могут быть использованы как фактические параметры при вызове функции SUMMA:

```
Sf := SUMMA (0.001, F1);
Sg := SUMMA (0.00001, G2);
```

Примеры программ

Пример 1. Построить столбчатую диаграмму (гистограмму), отображающую динамику перевозки пассажиров по дням недели на маршруте.

Для ввода значений количества пассажиров, можно воспользоваться датчиком случайных чисел Random.

```
program Grafika_Diagr_Color;
    uses GraphABC;
    const dx=40;    {ширина столбика}
          z=5;          {расстояние между столбиками}
          k=7;          {число элементов массива}
      delta = 10;       {диапазон изменения переменной от -10о до 10о}
var
     x, y, color, i, t, xm, ym: integer;
     a: array [1..7] of integer;
     Colors: array [1..k] of integer;
begin
   TextOut(80,50,'количество пассажиров на маршруте за неделю');
   TextOut(30,70,'X');                        {вывод надписи  X}
   TextOut(400,185,'Y');                      {вывод надписи  y}
   xm:=WindowWidth;                           {ширина окна}
   ym:=WindowHeight;                          {высота окна}
```

```
   y:=ym div 2;                                  {высота эл.}
   x:=dx+z;                                      {шаг вывода эл. диаграммы}
   Line(x-z,y,xm-2*z,y);                         {ось OX}
   Line(x-z,z,x-z,ym-z);                         {ось OY}
   SetWindowCaption('Перевозка пассажиров');
  SetPenStyle(psClear);
  Colors[1]:=clWhite;
  Colors[2]:=clLightGray;
  Colors[3]:=clGray;
  Colors[4]:=clDarkGray;
  Colors[5]:=clBlack;
  Colors[6]:=clRed;
  Colors[7]:=clGreen;
   TextOut(x-4*z,y,'0');                         {начало координат}
     for i:=1 to k do                            {количество вычислений}
       begin
        SetBrushColor(Colors[i]);
          a[i]:= Random(10);                     {вычисленное значение функции}
          Rectangle(x,(y-10*a[i]),x+dx,y);       {вывод диаграммы}
          write(x,'  ',(y-10*a[i]),'  ',x+dx,'  ',y,' ','     ');
          x:=x+dx+z                               {шаг изменения x}
       end
end.
```

Результат работы программы Grafika_Diagr_Color приведен на рисунке 1.

Рисунок 1 Гистограмма отображающая количество пассажиров на маршруте по дням недели.

Пример 2. Построить график, отображающий динамику перевозки пассажиров по дням недели на маршруте.

Для ввода значений количество пассажиров (тыс. человек), можно воспользоваться датчиком случайных чисел Random.

```
program Grafika_Line;
    uses GraphABC;
    const dx=40;    {ширина столбика}
          k=7;          {число элементов массива}
      delta = 10;       {диапазон изменения переменной ДСЧ от -10о до 10о}
var
     x, y, color, i, t, xm, ym: integer;
```

```
     a: array [1..8] of integer;
begin
SetWindowSize(400,400);
   TextOut(70,80,'количество пассажиров на маршруте по дням недели');
   TextOut(150,90,'(тысяч человек)');
   TextOut(30,70,'X');                          {вывод надписи - X}
   TextOut(350,180,'Y');                        {вывод надписи - Y}
   xm:=WindowWidth;                             {ширина окна}
   ym:=WindowHeight;                            {высота окна}
   y:=ym div 2;                                 {высота эл.}
   x:=dx;                                       {шаг вывода эл. графика}
   Line(x,y,xm,y);                              {ось OX}
   Line(x,0,x,ym);                              {ось OY}
   TextOut(x-10,y,'0');                         {начало координат}
       for i:=1 to 8 do                         {Заполнение массива}
           a[i]:= Random(10);                   {исходными данными}
                  for i:=1 to k do              {цикл вывода  элементов}
       begin
          SetPenColor(clRed);                   {задание цвета графика}
          Line(x,(200-10*a[i]),x+dx,(200-10*a[i+1]));  {вывод элемента}
          x:=x+dx                                      {шаг изменения x}
       end
end.
```

Упражнения

1)Ввести в гистограмме подписи значений температуры под соответствующими столбцами.

2) Увеличить период наблюдений за перевозкой пассажиров до декады, месяца.

3) Заменить диапазон изменения количества на [–20o , +40o].

4) Выполнить заливку столбиков гистограммы цветом границы прямоугольника (красным или синим), используя процедуру заливки FloodFill библиотеки GraphABC (см. Приложение).

5) Объяснить, почему при заливке столбиков (упражнение 4)) в случае нулевой температуры ось OX, а иногда и ось OY, окрашивается цветом границы.

6)Выполнить заливку столбиков гистограммы цветами, отличными от цветов границ, например, темно–красным или темно–синим (см. Приложение).

7) Организовать в программе ввод значений количества пассажиров.

Пример 2 Построить график функции f(x) на отрезке [a, b]. Считать, что функция на отрезке определена и непрерывна. При построении графика использовать такой масштаб, чтобы отрезок [a, b] занимал все графическое окно по ширине. Точки, соответствующие наибольшему (ymax) и наименьшему (ymin) значениям функции, должны располагаться соответственно в верхней и нижней части графического окна.

Если значение x=0 принадлежит отрезку [a, b], то изобразить на графике ось OY. Если отрезок [ymin ,ymax] содержит значение 0, то изобразить на графике ось OX.

Наибольшую трудность при построении графика вызывает его масштабирование в границах графического окна. Координатные оси графического окна направлены слева – направо (ось OX) и сверху – вниз (ось OY). Графические координаты принимают только целые неотрицательные значения. Максимальные значения графических координат определяются размерами графического окна (шириной и высотой). Точка в левом верхнем углу графического окна имеет графические координаты (0, 0). Точка в правом нижнем углу – графические координаты (W, H), где W – ширина графического окна, H – высота графического окна.

Для определения графических координат xg и yg можно воспользоваться следующими формулами преобразования:

xg= round (x0+(x–a)*Mx),

где x∈[a, b], Mx=(xW – x0)/(b – a),

x0, xW – графические координаты границы рисунка графика;

yg= round(y0+(ymax–y)*My),

где y∈[ymin, ymax], My=(yH – y0)/(ymax – ymin),

y0, yH – графические координаты границы рисунка графика;

round (X) – функция округления вещественного X до ближайшего целого.

Координаты x0, xW, y0, yH могут быть установлены равными соответственно 0, W, 0, H – в этом случае график займет все графическое окно. Можно установить значения x0, xW, y0, yH такими, чтобы вокруг графика были отступы определенного размера.

```
program Grafika;
uses GraphABC;
        var a, b : real;
        n : integer;
function F(x:real):real;
    begin
        F:= x*x-4*abs(x)+3
    end;
procedure MaxMin(a, b, h : real; var ymin, ymax : real);
        var x,y:real;
     begin
        ymin:=F(a); ymax:=ymin;
        x:=a;
```

```
        while x < b+h/2 do
                begin
                y:=F(x);
                if y < ymin then
                ymin:=y;
                if y > ymax then
                ymax:=y;
                x:=x+h
                end
end;
procedure GrFunc(a, b : real; n : integer);
var ymin, ymax x, y, h, Mx, My : real;
    xg, yg, xgp, ygp, ox, oy,
    x0, y0, xW, yH, i : integer;
        function prx(x : real):integer;
        begin
                prx:=round(x0+(x-a)*Mx)
        end;
        function pry(y : real):integer;
        begin
                pry:=round(y0+(ymax-y)*My)
        end;
begin
  h:=(b-a)/n; MaxMin(a,b,h,ymin,ymax);
  x0:=0; xW:=WindowWidth;
  y0:=0; yH:=WindowHeight;
  Mx:=(xW-x0)/(b-a);
  My:=(yH-y0)/(ymax-ymin);                    { Выводятся оси }
  if ymin*ymax < 0 then
  begin
     oy:=pry(0);
     Line(x0,oy,xW,oy);
     TextOut( xW-15,oy+10,'Ox')
  end;
  if a*b < 0 then
  begin
     ox:=prx(0);
     Line(ox,y0,ox,yH);
     TextOut( ox+10,y0+15,'Oy')
  end;
  x:=a;y:=f(a);
  xgp:=prx(x); ygp:=pry(y);
  for i:=1 to n do
  begin
     x:=x+h; y:=f(x);
     xg:=prx(x); yg:=pry(y);
     Line(xgp,ygp,xg,yg);
     xgp:=xg; ygp:=yg
```

```
  end;
end;
begin
  n:=50;
  write ('Введите концы отрезка a..b ');
  readln(a,b);
  GrFunc(a,b,n)
end.
```