ОБРАБОТКА ИНФОРМАЦИИ В СОВРЕМЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Основные понятия языка TURBO PASCAL

Средства графического пакета языка TURBO PASCAL

г. Днепропетровск

СОДЕРЖАНИЕ

1. АЛФАВИТ ЯЗЫКА ПАСКАЛЬ

Алфавит языка Паскаль составляют:

1) **буквы латинского алфавита** от A до Z и от a до z;
2) **арабские цифры** от 0 до 9;
3) **шестнадцатеричные цифры –** арабские цифры от 0 до 9, буквы от A до F;
4) **разделители –** символ пробела (код ASCII 32) и все управляющие символы кода ASCII (ASCII 0-31), включая символ конца строки или символ возврата каретки (ASCII 13);
5) **специальные знаки** это:

   одиночные символы:

   + - * / = < > [ ] . , ( ) : ; ^ @ { } $ #

   и *пары символов:*

   <= >= := .. (* *) (. .)

2. ОСНОВНЫЕ ПОНЯТИЯ ЯЗЫКА

Идентификаторы

Идентификаторы – это имена операторов, переменных, констант, типов величин (и имя самой программы).

Правила записи идентификаторов:

- идентификатор должен быть уникальным;
- длина идентификатора не должна превышать 6-8 символов;
- идентификатор может состоять ***только из символов латинского алфавита, цифр и знака подчеркивания(_)*** и не может содержать пробелов;
- идентификатор не может начинаться с цифры;
- можно использовать как строчные, так и прописные буквы;
- некоторые имена зарезервированы в языке для использования как служебные, их нельзя использовать в качестве идентификаторов (например: *begin, program, end, for, read, sin, uses, const*).

Правильно	Неправильно
den1	1den
MinMax	min(max)
Alfa_1	Alfa 1

Константы

Константы – это величины, значение которых не может быть изменено в ходе выполнения программы.

Описание именованных констант начинается служебным словом Const. Далее следуют записи вида: <Идентификатор>=<значение>;

Пример: **Const** Pi=3.14;
Name1= 'Татьяна';
X=6.33187E+03;

Переменные

Переменная – именованный участок памяти для хранения данных определенного типа.

Конкретные переменные и константы представляют собой объекты уникальные и отличаются друг от друга именем. В качестве данных в программах на языке Паскаль могут выступать числа, символы, целые строки символов.

3. СТРУКТУРА ПРОГРАММЫ

Правила языка Паскаль предусматривают единую для всех программ структуру. Программа состоит из разделов:

- раздел объявления меток;
- раздел объявления констант;
- раздел объявления типов;
- раздел объявления переменных;
- раздел объявления процедур и функций;
- раздел инструкций программы.

Структура программы в общем виде выглядит следующим образом:

Program <Имя программы>
<*раздел описаний*>
Begin
< *тело программы*>
End.

Имя программы выбирается программистом самостоятельно в соответствии с правилами, принятыми в языке Паскаль для идентификаторов.

Раздел описаний – раздел программы, в котором должны быть описаны все объекты, не являющиеся зарезервированными Паскале.

Тело программы – секция исходного кода, основной блок, в котором описываются выполняемые действия. Начинается зарезервированным словом **Begin** и заканчивается **End**.

Структура раздела описаний.

Program <Имя программы>	Блок описания программы, имя программы – идентификатор. Пример: **Program start;**
Label < список имен меток через запятую >	Имя метки должно быть идентификатором или целым числом между 0 и 9999. Пример: **Label 1, 2, metka1;**
Const <Имя константы>=<Значение константы>	Раздел объявления констант. Пример: **Const** time=100;
Type <имя типа**>=<**область значений**>**	Раздел описания типов. Пример: **Type mytype=1..1000;**
Var <список имен переменных через запятую>:<тип переменной>	Раздел объявления переменных. Пример: **Var x: real; y,z: integer;**
Procedure <имя процедуры>; **Function** <имя функции>;	Раздел объявления процедур и функций программиста. **Procedure myProc;** **Function new;**

Пример программы

```
Program stroke;
Uses    Crt;                    {для доступа к процедуре Delay }
Const   time=50000;             {константа времени}
Var     msg: string[80];        {сообщение }
        n: integer;             {номер выводимого символа}
Begin
msg:='Приветствую великого программиста! ';
for n:=1 to Length(msg) do      {вывод на экран сообщения
  Begin                                посимвольно}
        write(msg[n]);          {вывод не экран 1 символа с номером n}
        Delay(time);            { задержка 0.5 сек }
  End;
readln;
End.
```

После каждого оператора ставится точка с запятой.
После **begin** не ставится точка с запятой, а после **end** стоит точка.

4. СТАНДАРТНЫЕ ТИПЫ ДАННЫХ

Тип переменной указывает на то, какие данные могут быть сохранены в этом участке памяти, и в каких действиях эта переменная может участвовать.

К основным типам относятся:

- тип целых чисел – Integer;
- тип "длинных" целых чисел - Longint;
- тип действительных (вещественных) чисел - Real;
- тип неотрицательных целых чисел от 0 до 255 - Byte;
- тип неотрицательных целых чисел от 0 до65535 - Word;
- символьный тип - Char;
- строковый тип - String;
- логический тип – Boolean.

Физически типы данных отличаются друг от друга количеством ячеек памяти (байтов), отводимых для хранения соответствующей переменной.

Целочисленный тип и действия над целыми величинами.

Тип		Диапазон значений
Короткое целое	**Shortint**	-128 .. 127
Целое	**Integer**	-32 768 .. -32 767
Длинное целое	**Longint**	-2 147 483 648 .. 2 147 483 647
Длиной в байт	**Byte**	0 .. 255
Длиной в слово	**Word**	0 .. 65 535

Для целых величин определены следующие операции и функции

Функция	**Значение**	**Пример**
+ - *	Сумма, разность, произведение	x:=(a+a)*c
div	Частное от деления	x:=5 div 2; (результат=2) x:=10 div 2; (результат=5)
mod	Остаток от деления	x:=5 mod 2; (результат=1) x:=10 mod 2; (результат=0)
sqr(x)	Квадрат числа, x^2	a:=sqr(x)+sqr(y); (x^2+y^2)
abs(x)	Абсолютная величина, |x|	y:=abs(x);
trunc(x)	Целая часть числа x	d:=trunc(4.6); (результат=4)

	(x- вещественное число)	x:=4.1; d:=trunc(x); (результат=4)
round(x)	Округление числа x (x- вещественное число)	d:=round(4.6); (результат=5) d:=round(4.1); (результат=4)
dec(N) **dec(N,d)**	Уменьшает число N на 1 или на число d (эквивалентно N:=N-d)	dec(a); dec(N,5)
inc(x) **inc(N,x)**	Увеличивает на 1 число N или на число d (эквивалентно N:=N+d)	inc(a), inc(N,2)

Вещественный тип

Тип	Формат	Диапазон значений	Количество значащих цифр
Вещественное	**Real**	$2.9*10^{-39}$ - $1.7*10^{38}$	11-12
С одинарной точностью	**Single**	$1.5*10^{-45}$ - $3.4*10^{38}$	7-8
С двойной точностью	**Double**	$5.0*10^{-324}$ - $1.7*10^{308}$	15-16
С повышенной точностью	**Extended**	$1.9*10^{-4951}$ - $1.1*10^{4932}$	19-20

Вещественные константы в Паскале изображаются

1) в обычной форме с десятичной точкой 3.141529; 0.5 ; +3.0

2) в экспоненциальной форме N=MEP, где 1<=|M|<10, P – число, E – означает «10 в степени».

Пример: $5.00000E-01 = 5*10^{-1} = 0.5$

123.456 = 1.23456E+02

250 = 25E+1 = 2.5E+2 = 0.25E+3 = 2500E-01 = 25000E-02

При выводе на экран числа оператором **Write(x)** используется по умолчанию экспоненциальная форма.

Для вещественных чисел определены следующие операции и функции

Функция	Значение	Пример
+ - *, /	Сумма, разность, произведение, деление	x:=(a+a)*c/2;
abs(x)	Абсолютное значение x	Y:=abs(X);
arctan(x)	Арктангенс x	Z:=arctan(x*2);
cos(x)	Косинус x	Y:=cos(x)+cos(z);

exp(x)	e^x	Y:=exp(10);
frac(x)	Дробная часть x	X:=2.3; y:=frac(x); (результат 3)
int(x)	Целая часть x	X:=2.3; y:=int(x); (результат 2)
ln(x)	Натуральный логарифм x	X:=ln(10);
pi	3.1415926535897932385	X:=pi/4;
sin(x)	Синус x	Y:=sin(x*2);
sqr(x)	Квадрат x	Z:=sqr(x)+sqr(y);
sqrt(x)	Корень x	Z:=sqrt(x*x+y*y);

Литерные типы

Символьная константа (или литерная) есть любой символ языка, заключенный в апострофы.

Например: ‘A’, ‘+’, ‘9’. Чтобы представить апостроф как символьную константу, его повторяют дважды: ‘’’’.

Символьная переменная (тип Char) – это переменная, принимающая значение символьной константы.

К символьным данным применимы операции сравнения <, <=, =, < >, >=, >. Например: 'A′ > ′W′.

Строковая переменная (тип String) – это переменная, принимающая значение строковой константы (строка символов).

Пример:

```
R: = ′ kadabra ′;
H: = ′abra′ ;
S: = H+ R;{значением переменной S будет ′ abrakadabra ′}
```

Для работы со строками используются следующие процедуры и функции.

Процедура	Описание
Concat	Выполняет конкатенацию (объединение) последовательности строк
Copy	Возвращает подстроку строки.
Delete	Удаляет из строки подстроку.
Insert	Добавляет в строку подстроку.
Lenght	Возвращает динамическую длину строки.
Pos	Производит поиск подстроки в строке.
Str	Преобразует численное значение в его строковое представление
Val	Преобразует строковое значение в его численное представление.

Логический тип

В языке ПАСКАЛЬ имеются две логические (булевы) константы: TRUE (истина) и FALSE (ложь). Логическая переменная принимает одно из этих значений и имеет тип BOOLEAN.

Для сравнения данных предусмотрены следующие операции отношений: < (меньше); <= (меньше или равно); = (равно); <>(не равно); >=(больше или равно) ;> (больше).

Если операцию отношения приложить к арифметическим данным, то получим логическое значение: отношение истинно или ложно. Например, отношение 5>3 (читается «пять равно трем?») дает ложный результат (FALSE).

Булевское выражение – это такое выражение, которое принимает значение True или False. Оно состоит из булевых переменных, операций отношений, булевских операций, и/или других булевых выражений.

Правила составления выражений

При составлении выражений следует выполнять следующие правила.

- Записывать все составные части выражений в одну строку. Верхние и нижние индексы не допускаются.
- Использовать скобки только одного типа – круглые. Применение фигурных и квадратных скобок в выражениях запрещается, так как они имеют особое назначение.
- В правильно записанном выражении число открывающихся скобок всегда должно равняться числу закрывающихся скобок.
- Нельзя записывать подряд два знака арифметических операций. Например, выражение 3*A*B/-Z неверно. Его следует записать так: 3*A*B/(-Z).

Правила записи стандартных функций

- Имя функции записывается прописными буквами латинского алфавита. Имя состоит не более чем из шести букв.
- Аргумент функции записывается в круглых скобках после имени функции.
- Аргументом функции может быть константа, переменная или арифметическое выражение.

Пример:

Формула	Запись по правилам Паскаля
$WAY = \sqrt{dx^2 + dy^2}$	WAY: = sqrt(dx*dx+dy*dy)
$\frac{0.25\,y^2 + \sin(y)}{10}$	y1: = (abs(y)+ sin(y))/10

Обратите внимание:

- В тригонометрических функциях синуса или косинуса аргумент (угол) может быть задан только в радианной мере.
- В языке ПАСКАЛЬ нет операции возведения в степень. При необходимости ее использования применяют умножение или стандартные функции. Т.е. выражение $\mathbf{a^x}$ заменяют выражением **exp** (x* **ln** (a)); где **exp** и **ln** – стандартные функции.

5. ОПЕРАТОРЫ

Операторы описывают те алгоритмические действия, которые должны выполняться.

Операторы бывают трех видов:

- простые
- составные
- структурные

Простой оператор - оператор, который не содержит в себе других операторов. К простым операторам в Паскале относятся операторы присваивания, операторы ввода-вывода, операторы процедуры и операторы перехода. Например: S: = 4*arctan(1)*sqr(R);

Составной оператор – это несколько операторов, объединенных программными скобками Begin – End.

Формат составного оператора:	Пример

Begin
<Оператор 1>;
…..
<Оператор N>;
End;
	Begin
	Z:=X;
	X:=Y;
	Write(X,Y,Z);
	End;

Структурные операторы строятся из других операторов, порядок выполнения которых должен быть последовательным (составные операторы и операторы над записями), определяемым условной передачей управления (условные операторы) или повторяющимся (операторы цикла).

Оператор присваивания

Оператор присваивания – основной оператор любого языка программирования. Общая форма записи оператора:

V:=A

Здесь V – имя переменной; «**:=**» - знак присваивания; A – выражение.

Оператор присваивания вычисляет значение выражения A, стоящего справа от знака операции присваивания :=, и присваивает полученное значение переменной V, стоящей слева.

Оператор присваивания применим не только к арифметическим, но и к логическим и символьным данным.

При использовании оператора присваивания необходимо, чтобы переменная в левой части и выражение в правой части оператора были одного и того же типа.

Пример:
```
Y:=A + round(B/3)*2;
SUM:= SUM + X;
C5:=2*K – sin(PI/4 - X).
```

Оператор перехода Goto

Это оператор, нарушающий прямолинейное выполнение программы и передающий управление в произвольную ее точку. Такая инструкция называется безусловным переходом

Goto <метка>;

Оператор, к которому происходит переход, должен быть помечен данной меткой.

Метка в стандарте языка Паскаль представляет собой целое неотрицательное число. Все используемые в программе метки должны быть перечислены в разделе описания меток, начинающимся служебным

словом **Label**. *Одной меткой можно пометить только один оператор*. Метка от помеченного оператора отделяется двоеточием.

Пример:

```
Program   f1;
Label     1, 2, 8;
Var       X, A,B: Integer;
Begin
2:        X:=A*B;
          Goto 2;
End.
```

Операторы ввода-вывода

Оператор ввода имеет вид:

Read (a_1, a_2 … a_n) или **Readln (a_1, a_2 … a_n),**

где a_1, a_2 … a_n – переменные, которые последовательно получают вводимые значения. Числовые значения указываются через пробел, признаком окончания ввода является нажатие клавиши **Enter.**

Оператор ввода **Readln(a_1,a_2…a_n)** сначала вводит значения a_1, a_2 … a_n, а затем осуществляет переход на новую строку. Этот один оператор равносилен использованию двух предыдущих операторов.

Допускается использование оператора ввода без параметров **Readln**, осуществляющего переход на новую строку при вводе данных.

Оператор вывода имеет вид:

Write(a_1, a_2 … a_n) , **Writeln (a_1, a_2 … a_n)** ,

где a_1, a_2 … a_n являются в простом случае либо переменными, либо строкой символов, заключенной в апострофы.

Пример:

```
Write ('Значение B=', B)      {Выводит на экран дисплея строку
                               ЗНАЧЕНИЕ B=,
                               а затем значение переменной B}
```

Формат вывода чисел указывается через двоеточие после переменной.

Для действительных чисел используется оператор вывода

Write (a : p : q).

Для вывода целых чисел формат дробной части не указывается

Write (a : p).

p – обозначает общее поле выводимого значения, которое включает в себя отрицательный знак числа или пробел для положительного числа, количество цифр в целой части, точку и количество цифр в дробной части; **q –** поле дробной части.

Пример:
```
Write ( Y:5:2);
Write ( Y:5:0);{дробная часть не выводится}
Write ( X:5 );
```

Операторы ветвления.

Условный оператор IF

Условные операторы позволяют выбрать для выполнения один из составных операторов (или не выбрать ни одного).

Условный оператор имеет вид:

If <условное выражение>
Then <оператор 1>
Else < оператор 2>;
	If <условное выражение>
	Then
	Begin
	<группа операторов 1>
	end
	Else
	Begin
	<группа операторов 2>
	End

If <условное выражение> **Then** <оператор 1>; (Краткая форма)	

Пример:
```
If      X< 1.5
Then                    {если X< 1.5, то}
        Z:= X+Y         {выполнить присваивание Z:= X+Y }
Else                    {иначе}
Z:= 1.5; {выполнить присваивание Z:= 1.5}
```

Условный оператор действует следующим образом.

1. Если результатом условного выражения является истинное значение (True), то выполняется оператор, следующий за ключевым словом Then.
2. Если результатом выражения является значение False и присутствует ключевое слово Else, то выполняется оператор, следующий за ключевым словом Else.

3. Если ключевое слово Else отсутствует, то выполняется оператор, следующий за условным оператором If.

В простых условиях могут применяться знаки операций сравнения (операций отношений):

- > (больше)
- < (меньше)
- = (равно)
- <> (не равно)
- >= (больше или равно)
- <= (меньше или равно)

При организации сложных условных выражений широко применяются логические функции.

Оператор выбор Case

Оператор выбора относится к структурным операторам и имеет следующую форму записи:

Case <выражение> **of**
<константа_1>: <оператор 1>;
<константа_2>: <оператор 2>;
…
<константа_n>: <оператор n>;
End;

Здесь **Case** (*в случае*), **of** (*из),* **End** – служебные слова.

Выражение может быть любым стандартным типом, кроме действительного (Real). Константа не может быть действительного типа.

Оператор выбора действует следующим образом:

- Если значение выражения равно одной из констант, то выполняется соответствующий ей оператор. Затем управление передается за пределы оператора выбора.
- Если значение выражения не совпадает ни с одной константой, то управление передается за пределы оператора выбора.

Пример.

```
Case K+1 of
 5:  Y:=sqr(X) ;
 11: Y:=sqrt(X) ;
 4:  Begin Z:=4*(A – B) ;Z:=Z+1; End;
 7:  Write(A, B)
End;
```

Если значение К+1 будет равно 5, то выполнится оператор присваивания Y:=sqr(X) и управление будет передано на оператор, расположенный после слова End. Аналогично, если значение К+1 будет равно 11, 4 или 7, то выполнится один соответствующий оператор и управление будет передано за пределы оператора выбора.

Переменная К должна быть объявлена как переменная целого типа. Кроме того, К, Х, А, В должны получить значения до выполнения оператора Case.

Применение логических выражений в операторе IF

Пример 1: Вычислить A=N+40, если значение N больше 15, но меньше 25. При всех других значениях N вычислить B=M+1. Условный оператор имеет вид:

```
If (N>15) AND (N<25)  Then A:=N+40
Else B=M+1;
```

If (N>15) AND (N<25) означает: когда выполняются одновременно оба условия N>15 и N<25, только тогда вычисляется значение A:=N+40.Значения переменных N и M определяют до выполнения оператора.

Пример 2: Если A>B, то нужно вычислить три оператора Y1=7, Y2=A, Y3= A+B. Если A≤ B, то нужно вычислить T1=2A и T2=A – B. Условный оператор имеет вид:

```
If A>B  Then Begin
                Y1:=7;
                Y2:=A;
                Y3:= A+ B
                End
Else     Begin
         T1:=2*A;
         T2:= A – B
         End;
```

Здесь используются два составных оператора. Возможны случаи, когда используются один составной оператор, а другой простой.

Операторы цикла

Цикл – это многократное повторение однотипных действий

Тело цикла – это те действия, которые нужно повторять.

Создатель языка Паскаль Никлаус Вирт расширил язык тремя специальными возможностями организации циклов: "Пока", "До", " С параметром". Они отличаются друг от друга, и каждый используется для своего класса задач.

Оператор цикла с предусловием

Этот оператор еще называется оператором цикла «ПОКА». Тело цикла будет выполняться, **пока истинно** условие цикла. Выход из цикла произойдет, когда условие перестанет выполняться.

На языке Паскаль структура цикла "Пока" записывается следующим образом:

While<условие**>Do<**оператор**>;**

Читается так: "Пока истинно условие, выполнять оператор".

Если необходимо выполнить несколько действий, то может быть использован составной оператор. Тогда формат оператора принимает такой вид:

```
While<условие>Do
Begin
   <оператор 1>;
   <оператор 2>;
   …..
End;
```

Если условие ложно изначально, то тело цикла не будет выполнено ни разу.

Если условие изначально истинно и в теле цикла нет действий, влияющих на истинность этого условия, то тело цикла будет выполняться бесконечное количество раз ("зациклится").

Оператор цикла с постусловием

Этот вид цикла отличается от предыдущего в основном тем, что проверка условия повторения тела цикла находится не перед ним, а после (цикл "До").

Каждая новая итерация (повторное выполнение тела цикла) происходит не тогда, когда условие справедливо, а как раз тогда, когда оно ложно.

В случае, когда условие цикла изначально истинно, тело цикла все равно будет выполнено хотя бы один раз.

Формат цикла на языке Pascal:

```
Repeat
      <оператор 1>;
      <оператор 2>;
      …..
Until <условие>;
```

Читается так: " Выполнять оператор 1, оператор 2, … до выполнения условия ".

Здесь не требуется использование составного оператора, потому, что сами слова Repeat и Until являются операторными скобками.

Оператор цикла с параметром

Иногда цикл с параметром называют "Для" или "For".
Форматов у этого вида циклов предусмотрено два :

For <параметр> **:= <**н.з.> **To <**к.з.> **Do <**оператор**>;**

For < параметр>**:= <**н.з.> **Downto <**к.з.> **Do <**оператор**>;**

Здесь Н.З. – начальное значение параметра, К.З. –конечное значение параметра.

В заголовке цикла используется переменная, называемая параметром. Значение параметра изменяется при каждой итерации цикла. Таким образом, задав начальное и конечное значения для такой переменной, можно точно установить количество выполнений тела цикла. В первом случае параметр увеличивается на единицу, во втором – уменьшается на 1.

В качестве переменной-параметра используется идентификатор переменной, которая должна иметь *перечислимый тип*(Integer, Char). Начальное и конечное значения должны иметь тип, совместимый по присваиванию с перечислимым типом.

Выполняется этот цикл по следующему алгоритму.

1. Переменной – параметру присваивается начальное значение.
2. Выполняется тело цикла.
3. Параметр автоматически изменяется на 1.
4. Если параметр превышает конечное значение, то происходит выход из цикла, иначе переход к пункту 2.

При использовании **Downto** параметр автоматически уменьшается на 1, а выход из цикла происходит тогда, когда параметр становится меньше конечного значения.

Таким образом, в отличие от первых двух видов цикла этот цикл используется тогда, когда известно необходимое количество выполнений тела цикла.

Сравнительная характеристика операторов цикла

While	Repeat	For
1) До начала цикла должны быть сделаны начальные установки переменных .		1)Начальная установка не требуется
2) В теле цикла должны присутствовать операторы, изменяющие переменные из условия так, чтобы цикл через некоторое число операций завершился (т.е.условие должно стать истинно или ложно)		2) Изменения в теле цикла значений переменных, стоящих в заголовке (переменная-параметр) не допускается.
3)Цикл работает, пока выполняется условие True.	3)Цикл работает, пока выполняется условие False.	3)Количество итераций в цикле неизменно и определяется значениями нижней и верхней границ значений параметра , а также шага цикла.
4)Цикл завершается, когда условие становится ложным.	4)Цикл завершается, когда условие становится истинным.	4) Нормальный ход работы цикла может быть нарушен оператором goto.
5) Цикл может не выполняться ни разу, если исходное значение условия при входе в цикл равно False.	5) Цикл обязательно выполняется как минимум один раз.	5)цикл может не выполняться ни разу, если шаг цикла будет изменять значения счетчика нижней границы в направлении, противоположном верхней границе.
6) Если в теле цикла требуется более одного оператора, то необходимо использовать операторные скобки (begin - end) .	6) Независимо от количества операторов в теле цикла использование операторных скобок не требуется.	6) Если в теле цикла требуется более одного оператора, то необходимо использовать операторные скобки (begin - end) .

Примеры использования операторов цикла.

Пример:

Найти сумму квадратов всех натуральных чисел от 1 до 100.

Решение с использованием цикла "Пока".

```
Program  Ex1;
Var
    A: Integer;
    S: Longint;
Begin
    A:=1; S:=0;
  While A<=100 Do
  Begin
    S:= S+A*A;
```

```
            A:= A+1
          End;
          Writeln(S)
        End.
```

Решение с использованием цикла "До"

```
          Program Ex2;
          Var
            A: Integer;
            S: Longint;
        Begin
            A:=1; S:=0;
          Repeat
            S:= S+A*A;
            A:= A+1
          Until A>100;
           End;
          Writeln(S)
        End.
```

Решение с использованием цикла " С параметром".

```
          Program Ex3;
          Var
            A: Integer;
            S: Longint;
        Begin
            S:=0;
          For A:=1 To 100 Do
            S:= S+A*A;
          Writeln(S)
        End.
```

6. СРЕДСТВА ГРАФИЧЕСКОГО ПАКЕТА

Модуль **Graph** языка Паскаль содержит типы, константы, переменные и подпрограммы, позволяющие программисту создавать изображения с использованием широкого набора графических адаптеров.

Для работы с графикой в реальном режиме используется библиотека **TURBO.TPL** и модуль **GRAPH.TPU**.

Кроме того, при работе с графикой используются графические драйверы – программы, обеспечивающие работу с графическим адаптером компьютера, и файлы шрифтов. Все драйверы видеоадаптеров находятся в файлах с расширением **.BGI** (Borland Graphics Interface – графический интерфейс фирмы Borland), например **GGA.BGI; EGAVGA. BGI** и др.

Тип драйвера обязательно должен соответствовать типу адаптера.

Подключение и инициализация графического модуля.

- Подключение графического модуля осуществляется с помощью операции **Uses Graph.**
- Программа начинается с обращения к процедуре **InitGraph ,** которая идентифицирует графическую аппаратуру и производит загрузку и инициализацию соответствующего графического драйвера, переводит систему в графический режим, а затем возвращает управление вызывающей программе.
- Вызов процедуры **CloseGraph** восстанавливает первоначально обнаруженный видеорежим (**InitGraph**) и освобождает память, используемую графическим драйвером.
- С помощью программ **RestoreCrtMode** и **SetGraphMode** можно переключаться между текстовым и графическим режимом.
- Внутренние ошибки модуля Graph возвращаются функцией **GraphResult,** которая возвращает код ошибки, показывающий состояние последней графической операции.

Структура графического экрана.

При работе с графикой весь экран разбивается на отдельные ″точки″ пиксели, которые можно закрасить в тот или иной цвет. Каждый пиксель имеет две координаты: X и Y. Координата X увеличивается по горизонтали слева направо, начиная от нуля, координата Y увеличивается по вертикали сверху вниз, также начиная от нуля. Таким образом, левый верхний пиксель имеет координаты (0,0). Количество пикселей зависит от типа адаптера и режима его работы.

Так же как и в текстовом режиме при использовании модуля **Crt**, модуль **Graph** позволяет выделять окна на экране дисплея. Графические процедуры и функции в этом случае используют координаты в пределах окна, причем левый верхний угол окна получает координаты (0,0).

В отличие от текстового режима в графическом режиме курсор, определяющий место на экране, с которого начинается изображение фигуры или текса, невидим, однако его можно переместить в любую точку окна экрана, посмотреть значения координат курсора и.т.д.

Пример начала программы:

```
Program GraphTest;
Uses Graph;                                  {подключение модуля Graph}
Var GraphDriver: Integer;
     GraphMode: integer;
     ErrorCode: integer;                     {задается автоматическое
Begin                                        распознавание драйвера графи-
   GraphDriver:=Detect;                      ческого адаптера}
   InitGraph(GraphDriver,GraphMode,′c:\tp\bgi′);  {инициализация графического
   ErrorCode:= GraphResult;                  режима, драйверы находятся в
   If ErrorCode<> grOk then                  каталоге c:\ tp\bgi}
        Begin                                {код ошибки будет сохранен
        Writeln(′Ошибка графики:′ Graph Er-   для выдачи сообщения об
              rorMsg(Error Code )) ;         ошибке}
        Writeln(′Программа аварийно завершила {выдача сообщения об ошибке с
        работу…′)                            указанием ее кода}
   Halt(1) ;                                 {останов программы}
End.
```

Основные графические операторы.

ClearDevice	Очистка экрана
SetColor(Color)	Цвет пера
SetBkColor(Color)	Цвет фона
SetLineStyle(lstyle,Pat,width)	Стиль линии lstyle=(0 – сплошная, 1 – ..... , 2 – _ . _ . _ . .) width = (0-норм, 3 – тройная)
SetFillStyle(zalivka,svet)	Стиль заливки для фигур zalivka=0 – нет заливки(цвет фона) 1 – сплошная заливка 7 – клетка 2 – -------- 8 – косая клетка 3 – ///////// 9 – гориз.линии 4 – ///// (жирные) 10 – редкие точки 5 – \\\\\ (жирные) 11 – частые точки 6 – \\\\\ (жирные) 12 –стиль определен пользователем
SetFillPattern(Pattern;Color)	Задает произвольный орнамент, а также цвет для заполнения фигур. Pattern – задаваемый пользователем орнамент, Color – цвет заполнения фигур (номер цвета в палитре).
Элементы изображений формируются с помощью процедур	
PutPixel (x, y, Color)	Точка цвета Color
Line(x1, y1,x2, y2)	Линия от точки (x1, y1) к точке (x2, y2)
LineTo(x2, y2)	Линия от текущей точки к точке с координатами (x2, y2)
Rectangle (x1, y1, x2, y2)	Прямоугольник **(x1,y1)** **(x2,y2)**
Bar(x1, y1, x2, y2)	Прямоугольник залитый цветом фона SetFillStyle(5,11) SetBkColor(11) ; Bar(x1, y1, x2, y2**)**
Circle(x, y, R)	Круг с координатами центра **(x, y)** и радиусом **R**
Ellipse(x, y, nach_ugol, Kon_ ugol, xRad, yRad)	Эллипсный сектор с координатами центра **(x, y),** радиусами **xRad, yRad,** nach_ugol и Kon_ ugol-начальный и конечный углы эллипсного сектора; Если : nach_ugol=0, Kon_ ugol=360- **эллипс**; x Rad =yRad - **круг**; xRad>yRad - **эллипс**;

FillEllipse(x, y, xR, yR)	Эллипс, залитый текущим цветом; если xR=yR – это круг (залитый), FillEllipse(x, y,3, 3) – точка
FloodFill(x, y, Border)	Заполняет текущим цветом вокруг точки (x, y) до границы заданного цвета Border SetColor(3) ; Circle(320, 240, 50) ; FloodFill(320, 240, 3)
DrawPoly(Numpoints;Polypoint)	Контур многоугольника Numpoints –число вершин многоугольника; Polypoints – переменная без типа, содержащая Numpoint+1 пар координат вершин многоугольника (координаты должны быть целого типа, перечисляться в той же последовательности, как они идут по контуру, причем первая вершина должна быть повторена и в конце перечисления).
FillPoly (Numpoints;Polypoints);	Многоугольник, закрашенный текущим орнаментом и цветом заполнения.
Вывод текста	
SetTextStyle(Font,direction,Size)	SetTextStyle(0,0,1) Font – размер, direction (0-горизонт.текст, 1 – вертикальный), Size - толщина
OutTextXY(x,y,'текст')	Вывод текста, начиная с координат (x,y). OutTextXY(200,300,'Нажмите клавишу')
Вывод числа	Str(radius,text); OutTextXY(200,300,'Нажмите клавишу')

Константы цвета

Код цвета	Название		Код цвета	Название	
0	**Black**	черный	**8**	**DarkGray**	темно-серый
1	**Blue**	синий	**9**	**Light Blue**	ярко-синий
2	**Green**	зеленый	**10**	**Light Green**	св.зеленый
3	**Cyan**	голубой	**11**	**Light Cyan**	ярко - голубой
4	**Red**	красный	**12**	**Light Red**	розовый
5	**Magenta**	фиолетовый	**13**	**Light Magenta**	малиновый
6	**Brown**	коричневый	**14**	**Yellow**	желтый
7	**LightGray**	св.серый	**15**	**White**	Белый
128	**Blink**	мерцание			

Применение циклов и процедур в графических программах.

Циклы в графических программах дают возможность организовать быструю смену графических образов, т.е. вызывать на экране монитора динамические эффекты графики.

Например, циклическая закраска и гашение точек, выбранных случайным образом, создает эффект их мерцания.

{"Небо в алмазах" - генератор точек с гашением в их окрестности набора точек}

program STAR	
Uses Crt, Graph;	
Var Gx, Gy, Gd, Gm, x, y, i: integer;	*{ Gd – код драйвера, Gm - код графического режима, Gx, Gy – координаты углов экрана, x, y – текущие координаты точек, i – код цвета }*
begin	
Gd:=detect;	*{код драйвера графического устройства определяется автоматически}*
InitGraph(Gd, Gm,'c:/tp/bgi');	*{инициализация графики, драйвер находится в каталоге c:/tp/bgi }*
Gx:=GetMaxX;	*{максимальная координата по горизонтали}*
Gy:=GetMaxY;	*{максимум по вертикали}*
randomize;	*{вызывается процедура генерации случайных чисел}*
Repeat	*{цикл повторяется до нажатия любой клавиши}*
i:=random(7)+9;	*{случайное значение от 9(светло-синий) до 15(белый)}*
x:=random(Gx) ;	*{x – случайное значение от 0 до Gx , т.е.до максимума по горизонтали}*
y:=random(Gy) ;	*{то же по вертикали}*
Putpixel (x, y, i) ;	*{пиксел с координатами (x, y) закрашивается цветом с кодом i}*
for i:=0 to random (900) do	
Putpixel (x – 15 + random(30), y-15+ random(30), 0)	*{гашение набора точек в окрестности x, y }*
Until KeyPressed;	*{до нажатия любой клавиши}*
Read Key;	
Cleardevice;	*{очистка активной страницы}*
CloseGraph	
end.	

7. ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ МОДУЛЯ CRT.

Модуль Crt содержит константы, переменные и подпрограммы, предназначенные для работы с консолью. В отличие от стандартного ввода-вывода, когда он осуществляется через операционную систему, *подпрограммы модуля Crt работают с BIOS, и даже непосредственно с видеопамятью.*

ClrScr	Очищает текущее окно, заполняя его цветом фона и помещает курсор в его верхний левый угол с координатами (1,1). Цвет фона задается процедурой TextBackground.
Delay(Ms)	Задает задержку выполнения программы в Ms миллисекунд.
Goto XY(X, Y).	Перемещает курсор к элементу экрана с заданными координатами. X,Y – координаты элемента экрана (координаты отсчитываются от левого верхнего угла текущего окна).
HighVideo	Устанавливает высокую яркость символов (заменяет цвета 0-7 на цвета 8-15), выводимых далее на экран.
LowVideo	Устанавливает малую яркость символов (заменяет цвета 8-15 на цвета 0-7), выводимых далее на экран.
NormVideo	Устанавливает первоначальную яркость символов, выводимых далее на экран.
NoSound	Выключает источник звука. Источник звука включается процедурой Sound.
Sound(Hz)	Запускает источник звука с частотой Hz герц. Источник звука отключается процедурой NoSound. Hz частота звука в герцах.
TextBackground(Color)	Задает цвет фона. Color задаваемый цвет фона.
TextColor(Color)	Задает цвет символов Color.
Window (X1, Y1, X2, Y2: Byte);	Задает размеры окна на экране и помещает курсор в левый верхний угол окна с координатами (1, 1). X1, Y1 – координаты правого нижнего угла окна; X2, Y2 – координаты правого нижнего угла окна. Если хотя бы одна из координат недопустима, процедура не выполняется.

Функции.

KeyPressed	Анализирует нажатие клавиши клавиатуры (за исключением вспомогательных клавиш Shift, Alt, NumLock и.т.п.). Результат True или False, если клавиша на клавиатуре нажата или нет).
ReadKey	Считывает символ с клавиатуры и освобождает буфер клавиатуры от считанного символа.
WhereX	Возвращает текущую координату X курсора.
WhereY	Возвращает текущую координату Y курсора.