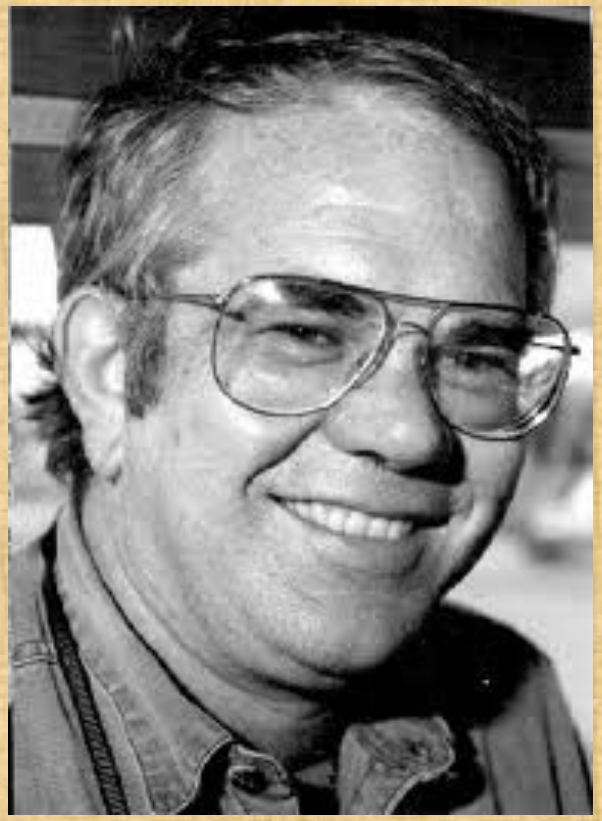
Турбо Паскаль



Никлаус Вирт

Блез Паска́ль

Основные понятия системы программирования Турбо Паскаль

Ввод и вывод данных

Ввод данных – это передача информации от внешнего носителя в оперативную память для обработки.

Вывод данных - обратный процесс, когда данные передаются после обработки из оперативной памяти на внешний носитель (экран монитора, принтер, дискету или винчестер и другие устройства).

Выполнение этих операций производится путем обращения к стандартным процедурам: ***Read, Readln, Write, Writeln.***

Основные понятия системы программирования Турбо Паскаль

Ввод данных с клавиатуры

Процедура чтения Read обеспечивает ввод данных для последующей их обработки программой.

Общий вид: ***Read (<список переменных>);***

В списке перечисляются имена переменных. Значения этих переменных набираются через пробел на клавиатуре и высвечиваются на экране после запуска программы.

Процедура чтения ReadIn аналогична процедуре Read, единственное отличие в том, что после считывания последнего в списке значения курсор переходит на начало новой строки.

Пример:

```
Program primer;  { Имя программы }
Var   i, k:integer;  c,d, s: real; { Описание
                          переменных}
begin
readln (c,d);  { Ввод c и d }
read(i,k);  {Ввод i и k }
…
end.
```

Основные понятия системы программирования Турбо Паскаль

Вывод данных

Процедура вывода Write производит вывод данных.
Общий вид: ***Write(<список вывода>);***

В списке вывода могут быть представлены выражения допустимых типов данных (integer, real, char и т.д.) и произвольный текст, заключенный в апострофы.
Например, ***Write('Привет'); Write(34.7); Write(45+55); Write(b, d);***

В процедурах вывода Write и Writeln имеется возможность записи выражения, определяющего ширину поля вывода.

При рассмотрении форматов вывода примем следующие обозначения: I – целочисленное выражение;
R - выражение вещественного типа; _ - пробел.

Значение I	Выражение	Результат
324	Write (I);	324
34	Write (I, I, I);	343434
324	Write (I : 6);	___324
312	Write (I + I : 7);	____624
Значение R	Выражение	Результат
123.432	Write (R);	__1.2343200000E+02
-1.34E+01	Write (R);	_ -1.3400000000E+01
304.55	Write (R :15);	3.045500000E+02
Значение R	Выражение	Результат
304.66	Write (R :8 : 4);	304.6600
45.322	Write (R : 5 : 2);	45.32

Пример:

```
Program primer;
  Var
  a, b, c, sum:integer;
begin
a:=4; b:=6; c:=55;
Write(a:3); Write(b:3); Write(c:3);
Sum:=a+b+c;
Writeln (‘A=’,a);
Writeln (‘B=’,b);
Writeln (‘C=’,c);
Writeln (‘Сумма A+B+C равна ’, sum);
End.
```

Операторы языка Паскаль

Общие сведения

Оператором называется предложение языка программирования, задающее полное описание некоторого действия, которое необходимо выполнить.

Разделителем операторов служит точка с запятой. Операторы, не содержащие других операторов, называются простыми. К ним относятся *операторы присваивания, безусловного перехода, вызова процедуры, пустой.*

Структурные операторы представляют собой конструкции, построенные из других операторов по строго определенным правилам. Эти операторы можно разделить на три группы: *составные, условные и повтора.*

Операторы языка Паскаль

Оператор вызова процедуры

Оператор вызова процедуры служит для активизации стандартной процедуры или процедуры, определенной пользователем.

Стандартные процедуры находятся в файлах, подключаемых модулем и для их использования достаточно указать имя процедуры, и если необходимо дополнительные параметры.

Для того, чтобы вызвать свою процедуру, ее для этого надо описать перед началом программы (begin), а затем уже использовать.

Например, ***ClrScr;*** {Вызов стандартной процедуры очистки экрана}.

Операторы языка Паскаль

Оператор безусловного перехода

Оператор безусловного перехода (***go to***) означает «перейти к» и применяется в случаях, когда после выполнения некоторого оператора надо выполнить не следующий по порядку, а какой-либо другой, отмеченный меткой, оператор.

Общий вид: ***go to <метка>.***

Имя метки может содержать цифровые и буквенные символы, максимальная длина имени ограничена 127 знаками. Раздел описания меток начинается зарезервированным словом ***Label***, за которым следует имя метки.

Пример.

```
Program primer;
Label 999, metka;
Begin
….
Go to 999;
…
999: write (‘ Имя’);
…
Go to metka;
….
Metka: write(‘Фамилия’);
…
end.
```

Рекомендуется минимальное использование оператора безусловного перехода с соблюдением следующих правил:

- Следует стремиться применять операторы перехода для передачи управления только вниз (вперед) по тексту программы;
- Расстояние между меткой и оператором перехода на нее не должно превышать одной страницы текста (или высоты экрана дисплея).

Операторы языка Паскаль
Пустой оператор

Пустой оператор не содержит никаких символов и не выполняет никаких действий. Используется для организации перехода к концу блока в случаях, если необходимо пропустить несколько операторов, но не выходить из блока. Для этого перед зарезервированным словом end ставятся метка и двоеточие, например:

```
Label m;
…
begin
…
go to m;
…
m:
end;
```

Операторы языка Паскаль
Структурные операторы

Структурные операторы представляют собой конструкции, построенные из других операторов по строгим правилам. Их можно разделить на три группы: ***составные, условные и повтора***.

Применение структурных операторов в вашей программе очень часто просто незаменимо, потому что они позволяют программисту сделать его программу зависимой от каких-либо условий, например введенных пользователем. К тому же применяя операторы повтора вы получаете возможность обрабатывать большие объемы данных за сравнительно малый отрезок времени (это конечно же зависит и от процессора)

Операторы языка Паскаль
Составной оператор

Это оператор представляет собой совокупность произвольного числа операторов, отделенных друг от друга точкой с запятой, и ограниченную операторными скобками ***begin*** и ***end***.

Он воспринимается как единое целое и может находиться в любом месте программы, где возможно наличие оператора.

Операторы языка Паскаль
Условные операторы

Условные операторы предназначены для выбора к исполнению одного из возможных действий, в зависимости от некоторого условия (при этом одно из действий может отсутствовать).

Для программирования ветвящихся алгоритмов в Турбо Паскале есть специальные операторы.

Одним из них является условный оператор ***If***.

Он может принимать одну из форм:

If <условие> then <оператор1>
else<оператор2>;

или

If <условие> then <оператор>;

Обратить внимание, что перед словом else точка с запятой не ставится.

Пример 1. Составить программу, которая запрашивает возраст ребенка и затем выдает решение о приеме ребенка в школу (возраст ≥ 7лет).

```
Program sh;
Var v: integer;
Begin
Write('Введите возраст ребенка');
Readln(v);
If v>=7 then writeln('Принимаем в школу')
        else writeln ('Не принимаем в школу');
end.
```

Задание. Модифицировать данную программу, чтобы ограничить верхнюю границу приема в школу 16 годами.

Решение:

```
Program sh;
Var v: integer;
Begin
Write('Введите возраст ребенка');
Readln(v);
If (v>=7) and (v<=16) then writeln('Принимаем в школу')
        else writeln ('Не принимаем в школу');
end.
```

Пример 2. Даны два числа. Меньшее из этих чисел заменить суммой данных чисел, большее - произведением.

```
Program sh;
Var x, y,s,p: integer;
Begin
Write('Введите 2 числа');
Readln(x, y);
S:=x+y; p:=x*y;
If x>=y
then begin y:=s; x:=p; end
else begin x:=s; y:=p; end;
writeln('x=', x);
writeln('y=', y);
end.
```

Если оператор ***If*** обеспечивает выбор из двух альтернатив, то существует оператор, который позволяет сделать выбор из произвольного числа вариантов.

Это оператор выбора ***Case***. Он организует переход на один из нескольких вариантов действий в зависимости от значения выражения, называемого селектором.

Общий вид: Case k of

<const1>: <оператор1>;
<const2>: <оператор2>;
…..
<constN>: <операторN>
else <операторN+1>
end;

Здесь k – выражение-селектор, которое может иметь только простой порядковый тип (целый, символьный, логический). <const1>, …<constN> - константы того же типа, что и селектор.

При использовании оператора Case должны выполняться следующие правила:

1. Выражение-селектор может иметь только простой порядковый тип (целый, символьный, логический).
2. Все константы, которые предшествуют операторам альтернатив, должны иметь тот же тип, что и селектор.
3. Все константы в альтернативах должны быть уникальны в пределах оператора выбора.

Формы записи оператора:

Селектор интервального типа:

```
Case I of
1..10    : writeln(‘число в диапазоне 1-10’);
11.. 20 : writeln(‘число в диапазоне 11-20’);
else writeln(‘число вне пределов нужных диапазонов’)
end;
```

Селектор целого типа:

```
Case I of
1 : y:= I+10;
2 : y:= I+20;
3:  y:= I +30;
end;
```

Пример 1. Составить программу, которая по введенному номеру дня недели выводит на экран его название.

```
Program days;
Var day:byte;
Begin
Write('Введите номер дня недели');
Readln(day);
Case day of
1: writeln('Понедельник');
2: writeln('Вторник');
3: writeln('Среда');
4: writeln('Четверг');
5: writeln('Пятница');
6: writeln('Суббота');
7: writeln('Воскресенье')
else writeln('Такого дня нет');
end; end.
```

Пример 2. Составить программу, которая по введенному номеру месяца выводит на экран название времени года.

```
Program m;
Var k:byte;
Begin
Write('Введите номер месяца');
Readln(k);
Case k of
1, 2, 12: writeln('Зима');
3, 4, 5: writeln('Весна');
6, 7, 8: writeln('Лето');
9, 10, 11: writeln('Осень')
else writeln('Такого месяца нет');
end; end.
```

Операторы языка Паскаль
Операторы повтора (цикла)

В языке Паскаль различают три вида операторов цикла: ***цикл с предусловием (while)***, ***цикл с постусловием (repeat) и цикл с параметром (for).***

Если число требуемых повторений заранее известно, то используется оператор, называемый оператором цикла с параметром.

Оператор цикла с параметром имеет два варианта записи:

1) for <имя переменной> := <начальное значение> to <конечное значение> do <тело цикла>
2) for <имя переменной> := <начальное значение> downto <конечное значение> do <тело цикла>

При первом обращении к оператору for вначале определяются начальное и конечное значения, и присваивается параметру цикла начальное значение. После этого циклически повторяются следующие действия.

1. Проверяется условие параметр цикла <= конечному значению.
2. Если условие выполнено, то оператор продолжает работу (выполняется оператор в теле цикла), если условие не выполнено, то оператор завершает работу и управление в программе передается на оператор, следующий за циклом.
3. Значение параметра изменяется (увеличивается на 1 или уменьшается на 1).

Если в теле цикла располагается более одного оператора, то они заключаются в операторные скобки begin … end;

Пример 1. Вывести на экран таблицу перевода из градусов по шкале Цельсия в градусы по Фаренгейту для значений от 15°C до 30°C с шагом в 1°C. Перевод осуществляется по формуле: F=C*1.8+32.

```
 Program zf;
Var i:integer; f:real;
Begin
Writeln('Температура');
For i:=15 to 30 do
Begin
F:=i*1.8+32;
Writeln('по Цельсию', i, 'по Фаренгейту', f:5:2);
End;
End.
```

Пример 2. Вывести на экран натуральные числа от 1 до 9 в обратном порядке.

```
Program z;
Var i:integer;
Begin
For i:=9 downto 1 do
Writeln(i);
End.
```

Если число повторений заранее неизвестно, а задано лишь условие его повторения (или окончания), то используются операторы ***while u repeat.***

Оператор While часто называют оператором цикла с предусловием. Так как проверка условия выполнения цикла производится в самом начале оператора.

Общий вид: ***While <условие продолжения повторений> do***
<тело цикла>;

Тело цикла – простой или составной оператор или операторы. Если операторов в теле цикла несколько, то тело цикла заключается в операторные скобки ***begin…end.***

Пример. Найти сумму 10 произвольных чисел.

```
Program z;
Const
N=10;
Var k, x, s: integer;
Begin
k:=0; s:=0;  {k- количество введенных чисел}
while k < n do
   begin
    k:=k+1;
    write('Введите число');
     readln(x);
     s:=s+x;
    end;
writeln('Сумма чисел равна', s);
end.
```

Оператор цикла ***repeat*** аналогичен оператору ***while***, но отличается от него, во-первых, тем, что условие проверяется после очередного выполнения операторов тела цикла и таким образом гарантируется хотя бы однократное выполнение цикла. Во-вторых, тем, что критерием прекращения цикла является равенство выражения константе true .

За это данный оператор часто называют циклом с постусловием, так как он прекращает выполняться, как только условие, записанное после слова until, выполнится. Оператор цикла repeat состоит из заголовка, тела и условия окончания. Общий вид: ***Repeat***

<оператор>

. . . . .

<оператор>

until <условие окончания цикла>

Пример. Составить программу, которая вводит и суммирует целые числа. Если введено значение 999, то на экран выводится результат суммирования.

```
Program s;
Var x, s:integer;
Begin
S:=0;
Repeat
Write('Ввести число');
Readln(x);
If x<>999 then s:=s+x;
Until x=999;
Writeln('Сумма введенных чисел', s);
End.
```

Служебные (зарезервированные) слова:

ABSOLUTE
ASSEMBLER
AND
ARRAY
ASM
ASSEMBLER
BEGIN
CASE
CONST
CONSTRUCTOR
DESTRUCTOR
DIV
DO
DOWNTO
ELSE
END
EXPORT
EXPORTS
EXTERNAL
FAR
FILE
FOR
FORWARD
FUNCTION
GOTO
IF
IMPLEMENTATION
IN
INDEX
INHERITED
INLINE
INTERFACE
INTERRUPT
LABEL
LIBRARY
MOD
NAME
NIL
NEAR
NOT
OBJECT
OF
OR
PACKED
PRIVATE
PROCEDURE
PROGRAM
PUBLIC
RECORD
REPEAT
RESIDENT
SET
SHL
SHR
STRING
THEN
TO
TYPE
UNIT
UNTIL
USES
VAR
VIRTUAL
WHILE
WITH
XOR

Стандартные операции и функции.

Операции бывают следующих видов:

- арифметические операции;
- операции отношений;
- булевские (логические) операции;
- поразрядные логические и сдвиговые операции;
- операции над множествами.

Арифметические операции:

«+» - сложение;

«*» - умножение;

«-» - вычитание;

«/» - деление; (результат всегда должен иметь вещественный тип).

div – деление нацело (с отбрасыванием дробной части);

mod – взятие остатка от целочисленного деления.

Стандартные математические функции

abs(x)	Абсолютное значение x, т.е. x
exp(x)	Значением функции является e в степени x.
sin(x) и cos(x)	Значение синуса или косинуса x, x должен задаваться в радианах.
arctan(x)	Арктангенс x.
ln(x)	Натуральный логарифм x (x>0)
sqr(x)	Квадрат x.
sqrt(x)	Квадратный корень из x.
random(x)	Случайное число из диапазона 0<=...< x
Pi	Значение пи.
odd(x)	Значение функции true, если x нечетен, и false в противном случае.
inc(x,n)	Значением является x увеличенное на n.
dec(x,n)	Значением является x уменьшенное на n.
int(x)	Целая часть числа x.
frac(x)	Дробная часть числа x.
trunc(x)	Целая часть в форме longint.
round(x)	Значение x округленное до следующего целого числа.

Правила записи арифметических выражений.

1. Все данные, входящие в арифметическое выражение, должны быть одного типа. Допускается использовать вместе данные целого и действительного типа.
2. Записывать все составные части в одну строку без подстрочных и надстрочных индексов.
3. Использовать скобки одного типа - круглые. ([{ и другие скобки применять запрещается)
4. Нельзя записывать подряд 2 знака арифметических операций.
5. Вычисления выполняются слева направо в соответствии со старшинством операций:

   1) вычисление функций;

   2) * / DIV (деление нацело) MOD (получение остатка от деления)

   3) + -

Правила записи стандартных функций.

1. Имя функции записывается латинскими буквами.
2. Аргумент функции записывается в круглых скобках после имени функции.
3. Аргументом функции может быть : константа, переменная или арифметическое выражение.

Например :

$ax^2 + bx + c$ записывают так a*x*x +b*x + c

$\sqrt{b^2 - 4ac}$ записывают так sqrt(b*b – 4*a*c)

$\frac{a + c - 2b}{3 - x}$ записывают так (a + c – 2*b)/(3-x)

Задание: Составить программу, которая по двум введенным с клавиатуры целым числам вычисляла бы и выводила на экран в удобном виде:

1. их сумму;
2. их произведение;
3. их разность;
4. их частное;
5. их среднее арифметическое;
6. сумму квадратов этих чисел;

с точностью до сотых.

Массивы

Массивом называется поименованная совокупность элементов с одинаковыми свойствами.

Массив не является стандартным типом данных, поэтому он задаётся в разделе описания типов:

type имя_типа = ***array*** [тип_индекса] ***of*** тип_элемента

array и ***of*** – ключевые слова, тип индекса задаётся в квадратных скобках.

Массивы бывают одномерные, двумерные, трёхмерные и т.д.

После имени массива, которое состоит из латинских букв (строчные, прописные – количество символов зависит от языка программирования), записывается в скобках (обычно в круглых) размерность массива, которая отображает внутреннюю структуру массива, т. е. взаимное расположение элементов по отношению к друг другу.

В записи размерности могут присутствовать переменные и цифры. Например:

A(I) , B(5) – одномерные массивы,

C(I,J), d(7,10), x(i,j) – двумерные массивы.

ОДНОМЕРНЫЕ МАССИВЫ
(линейные или вектора)

В ЭВМ одномерный массив представляет собой ряд последовательно расположенных поименованных ячеек, каждая из которых имеет свой номер (индекс). Номера ячеек в ЭВМ начинаются с нулевого значения, обычно обозначаются буквой латинского алфавита I (i), а для двумерного массива – I,J (i,j).

Рассмотрим одномерный массив A(5), который представляет собой 6 зарезервированных ячеек, заполним ячейки цифрами (ячейку с индексом 0 можно не использовать):

XXX	-3,5	10	0	0,017	-83
I= 0	1	2	3	4	5

A(1) = -3,5;

A(2) = 10;

A(3) = 0;

A(4) = 0,017;

A(5) = -83,

следовательно A(I) – элемент (число) массива, который находится в ячейке под номером (индексом) I.

Так как массив имеет одно имя и элементы массива отличаются индексами, то массив можно обрабатывать в цикле, изменяя значение индекса

ДВУМЕРНЫЕ МАССИВЫ (Матрицы)

Рассмотрим двумерный массив ***A(2,3)***, где ***2*** - количество строк. ***3*** - количество столбцов.

A(I,J) – элемент двумерного массива, находящийся в ячейке, (***I*** – индекс строки, ***J*** –индекс столбца), нулевые строку и столбец учитывать не будем

$$A(2,3) = \begin{vmatrix} A_{11} A_{12} A_{13} \\ A_{21} A_{22} A_{23} \end{vmatrix}$$

Свойства симметричной (квадратичной) матрицы (количество строк равно количеству столбцов)

Рассмотрим двумерный массив A(3,3)

$$A(3,3) = \begin{vmatrix} A_{11} A_{12} A_{13} \\ A_{21} A_{22} A_{23} \\ A_{31} A_{32} A_{33} \end{vmatrix}$$

Элементы $A_{11,}$ A_{22}, A_{33} – элементы главной диагонали, где I = J;

Элементы $A_{12,}$ A_{13}, A_{23} – элементы над главной диагональю, где I < J;

Элементы $A_{21,}$ A_{31}, A_{32} – элементы под главной диагональю, где I > J;

Элементы $A_{13,}$ A_{22}, A_{31} – элементы побочной диагонали, где J = 4 - I

Примеры описания типа:

mas = array [1..10] of real;

Color = array [byte] of mas;

Active = array [Menu] of Boolean;

Обычно при описании массива верхняя граница его индекса задаётся в виде именованной константы, например:

одномерный массив:

```
const n = 6;
type intmas = array [1..n] of integer;
```

двумерный массив:

```
const n = 3; m = 6;    (*m – количество строк, n – количество столбцов*)
type mas = array [1..n] of integer;
type mas2 = array [1..m] of mas;
```

или

```
type mas2 = array [1..m, 1..n] of integer;
```

После задания типа массива переменные этого типа описываются обычным образом, например:

```
var a, b : intmas;
```

или

```
var a, b : mas2;
```

Если требуемый тип массива используется только в одном месте программы, можно описать тип прямо при определении переменных:

var a, b : array [1..n] of integer;

С массивами можно выполнять только одну операцию: присваивание. При этом массивы должны быть одного типа, например:

b := a;

Все остальные действия выполняются с отдельными элементами массива, после имени массива указывается номер элемента в квадратных скобках, например:

a[4] b[i]

a[1, 4] b[i, j]

1. Найти максимальный элемент из 20 вещественных элементов

```
program max_element;
    const n = 20;
    var a   : array [1..n] of real ;                    {массив}
        i   : integer;                                  {номер текущего элемента}
        max   : real;                                   {максимальный элемент}
    begin
        writeln('Введите', n, 'элементов');
        for i := 1 to n do read(a[i]);
        for i := 1 to n do writeln(a[i]:6:2);           {отладочная печать}
        writeln;
        max := a[1];          {принять за максимальный первый  элемент массива}
        for i := 2 to n do    {просмотреть массив, начиная со второго элемента}
            if max < a[i]     {если очередной элемент больше максимального }
            then  max := a[i];      {принять за максимальный очередной элемент}
        writeln('Максимальный элемент', max:6:2);
        readln;
    end.
```