Операторы ввода — вывода данных

Ввод информации с клавиатуры осуществляется с помощью оператора read.

Он может иметь один из следующих форматов:

```
read ( x1 , x2 , . . . , xn ) ;
```

или

```
readln ( x1 , x2 , . . . , xn ) ;
```

где x1, x2,. . . , xn — список вводимых переменных.

При вводе вещественных значений целую и дробную часть числа следует разделять точкой.

Для вывода информации на экран служат операторы write и writeln. В общем случае эти операторы имеют вид:

```
write ( x1 , x2 , . . . , xn ) ;
```

или

```
writeln ( x1 , x2 , . . . , xn ) ;
```

где x1, x2,. . . , xn представляют собой список выводимых переменных, констант, выражений. Если элемент списка — текстовая информация, её необходимо взять в кавычки.

Чтобы выводить числа в формате с фиксированной точкой, необходимо использовать форматированный вывод. Для этого оператор write или writeln нужно задать следующим образом:

```
write (идентификатор :
ширина_поля_вывода :
дробная_часть ) ;
```

Оператор присваивания

В общем случае оператор присваивания имеет вид:

имя_переменной := значение ;

Сначала вычисляется значение выражения, указанного в правой части оператора, а затем его результат записывается в область памяти (переменную), имя которой указано слева. Например, запись a:=b означает, что переменной, а присваивается значение выражения b.

Стандартные функции

Обозначение	Тип результата	Тип аргументов	Действие
`abs(x)`	целый/вещественный	целый/вещественный	модуль числа
`sin(x)`	вещественный	вещественный	синус
`cos(x)`	вещественный	вещественный	косинус
`arctan(x)`	вещественный	вещественный	арктангенс
`pi`	без аргумента	вещественный	число π
`exp(x)`	вещественный	вещественный	экспонента e^x
`ln(x)`	вещественный	вещественный	натуральный логарифм
`sqr(x)`	вещественный	вещественный	квадрат числа
`sqrt(x)`	вещественный	вещественный	корень квадратный
`int(x)`	вещественный	вещественный	целая часть числа
`frac(x)`	вещественный	вещественный	дробная часть числа
`round(x)`	вещественный	целый	округление числа
`trunc(x)`	вещественный	целый	отсекание дробной части числа
`random(n)`	целый	целый	случайное число от 0 до n

Функции, определённые в программном модуле Math			
`arcos(x)`	вещественный	вещественный	арккосинус
`arcsin(x)`	вещественный	вещественный	арксинус
`arccot(x)`	вещественный	вещественный	арккотангенс
`arctan2(y,x)`	вещественный	вещественный	арктангенс y/x
`cosecant(x)`	вещественный	вещественный	косеканс
`sec(x)`	вещественный	вещественный	секанс
`cot(x)`	вещественный	вещественный	котангенс
`tan(x)`	вещественный	вещественный	тангенс
`lnXP1(x)`	вещественный	вещественный	логарифм натуральный от $x+1$
`log10(x)`	вещественный	вещественный	десятичный логарифм
`log2(x)`	вещественный	вещественный	логарифм по основанию два
`logN(n,x)`	вещественный	вещественный	Логарифм от x по основанию n

Условные операторы

Условный оператор if..then..else

Формат описания:

i f условие then оператор_1 else оператор_2 ;

Работа условного оператора организована следующим образом. Сначала вычисляется выражение, записанное в условии. Если оно имеет значение истина (True), то выполняется оператор_1. В противном случае, когда выражение имеет значение ложь (False), оператор_1 игнорируется и управление передаётся оператору_2.

Если в задаче требуется, чтобы в зависимости от значения условия выполнялся не один оператор, а несколько, необходимо использовать составной оператор:

```
i f условие then
begin
оператор_1 ;
оператор_2 ;
. . .
оператор_n ;
end
else
begin
оператор_1A ;
оператор_1B;
. . .
оператор_1N;
end ;
```

Альтернативная ветвь else в условном операторе может отсутствовать, если в ней нет необходимости:

```
i f условие then оператор ;
```

или

```
i f условие then
begin
оператор_1 ;
оператор_2 ;
. . .
оператор_n ;
end ;
```

Условные операторы могут быть вложены друг в друга. При вложениях условных операторов всегда действует правило: альтернатива else считается принадлежащей ближайшему if, имеющему ветвь else. Например, в записи

```
i f условие_1 then
i f условие_2 then
оператор_А
else оператор_Б ;
```

оператор_Б относится к условию_2, а в конструкции

```
i f условие_1 then
begin
i f условие_2 then
оператор_А ;
end
else оператор_Б ;
```

он принадлежит оператору if с условием_1.

Оператор варианта case

Оператор варианта case необходим в тех случаях, когда в зависимости от значений какой-либо переменной надо выполнить те или иные операторы.

```
case выражение of
значение_1 : оператор_1 ;
значение_2 : оператор_2 ;
. . .
значение_N: оператор_N
else
альтернативный_оператор;
end;
```

Альтернативная ветвь else может отсутствовать, тогда оператор имеет вид:

```
case выражение of
значение_1 : оператор_1 ;
значение_2 : оператор_2 ;
. . .
значение_N: оператор_N;
end;
```

Кроме того, в операторе case допустимо использование составного оператора.

Например:

```
case выражение of
значение_1 : begin оператор_A;
оператор_B; end;
значение_2 : begin оператор_C;
оператор_D; оператор_E; end;
. . .
значение_N: оператор_N;
end;
```

Пример

- Вывести символьное описание введенной цифры

```
Program Number1;
Var
  a : integer;
Begin
  writeln('Введите цифру ');
  readln(a);
  if (a<0) or (a>9)
    then
      writeln ('Это число не является цифрой')
    else
      case a of
        0 : writeln ('ноль');
        1 : writeln ('один');
        2 : writeln ('два');
```

```
Program Number2;
Var
  a : integer;
Begin
  writeln('Введите цифру ');
  readln(a);
  case a of
    0 : writeln ('ноль');
    1 : writeln ('один');
    2 : writeln ('два');
    3 : writeln ('три');
    4 : writeln ('четыре');
    5 : writeln ('пять');
```