Основные алгоритмические конструкции и типы данных.

Арифметика языка программирования Pascal ABC

Знаки арифметических действий в Паскале.

Запись знаков:

В математике	В Паскале
+ (сложение)	**+**
– (вычитание)	**–**
· (умножение)	*****
: (деление)	**/**
mod – остаток от целочисленного деления. Пример. 5 mod 3 = 2	
div – результат целочисленного деления Пример. 5 div 3 = 1	

Стандартные арифметические функций в Паскале.

1. **abs(a) –** $|a|$
2. **sqr(b) –** a^2
3. **trunc(a)** - целая часть числа a.
4. **round(a)** - округляет a до ближайшего целого.
5. **frac(a)** – дробная часть числа a.

Пример.

Если a=4.8
trunc(a) =4
round(a) =5
frac(a) =0.8

Внимание!!!

Дробная часть числа от целой в Паскале отделяется точкой.

Типы переменных после выполнения арифметических действий и функций в Паскале:

Операции и функции	Тип входных данных	Тип результата
+, -, *, abs, sqr	integer real	integer real
/	integer real	real real

div, mod	integer	integer
trunc, round	real	integer
frac	real	real

Задача 4. Даны 2 числа a и b. Найти сумму и произведение этих чисел и вывести результат на экран.

Внимание!!!

Если в условии задачи не указан тип чисел, то будем считать, что нам даны действительные числа (*real*).

Эту задачу можно решить так:

```
program z4;
var
    a,b,s,d:real;
begin
    read(a,b);
    s:=a+b;
    d:=a*b;
    writeln(s);
    writeln(d);
end.
```

```
program z4;
var
a,b,s,d:real;
begin
  read(a,b);
  s:=a+b;
  d:=a*b;
  writeln(s);
  writeln(d);
end.
```

Окно вывода

```
7 3
10
21
```

Можно решить и так:

```
program z4;
var
    a,b,s,d:real;
begin
    writeln('Введите 2 числа');
```

```
    read(a,b);
    s:=a+b;
    d:=a*b;
    writeln('s=',s);
    writeln('d=',d);
end.
```

```
program z4;
var
a,b,s,d:real;
begin
  writeln('Введите 2 числа');
read(a,b);
  s:=a+b;
  d:=a*b;
  writeln('s=',s);
  writeln('d=',d);
end.
```

Окно вывода

```
Введите 2 числа
7 3
s=10
d=21
```

Команда присваивания

В команде присваивания всегда слева от «:=» пишется только имя переменной, а справа может быть записано число либо любое арифметическое выражение.

Общий вид команды присваивания можно представить так:

<Имя переменной>:=<выражение>;

Запись математических выражений в Паскале имеет свои синтаксические правила и приоритеты вычислений. Например, нельзя писать выражения в виде обыкновенных дробей («многоэтажные» записи запрещены). Запись 4ac Паскаль тоже не понимает, операцию умножения опускать нельзя: следует писать 4*a*c. Выполнение каждой операции в выражениях происходит с учетом ее приоритета.

Порядок выполнения операций

1. Выражения в скобках вычисляются в первую очередь.
2. После вычисления значений выражений в скобках вычисляются все функции.
3. После функций выполняется умножение и деление (они имеют одинаковый приоритет и выполняются в порядке их следования слева направо).
4. Далее выполняется сложение и вычитание в порядке их следования.

Условный оператор. Простые и составные условия

В программировании можно создавать программы, умеющие выполнять выбор. Для этого существуют команды, которые позволяют компьютеру принимать решения в зависимости от выполнения некоторого условия. Одной из таких команд является условный оператор языка программирования Pascal

If *<условия>*
then begin
<набор операторов 1> ;
end
else begin
<набор операторов 2> ;
end;

В переводе на русский язык данная форма записи означает: если выполняются *условия*, то исполняется *набор операторов 1*, иначе исполняется *набор операторов 2*.

Рассмотрим на примере, как работает оператор if.

Например, необходимо дать задание компьютеру *проверить, знают ли ученики таблицу умножения.*

Задача 1. *Проверка знаний таблицы умножения*

```
program z1;
  var a:integer;
begin
  writeln('Сколько будет 3x5?');
  readln(a);                {запрашивается ответ, вводимый с клавиатуры,
                                  который записывается в переменную a}
  if a=15
   then begin               {оператор if проверяет и анализирует значение переменной
a.
    writeln('Верно');       Если a=15, то компьютер выводит сообщение «Верно»,
   end                      в противном случае сообщает «Неверно»}
   else begin
    writeln('Неверно');
   end;
end.
```

```
program z1;
    var a:integer;
begin
    writeln('Сколько будет 3x5?');
    readln(a);
    if a=15
      then begin
       writeln('Верно');
      end
      else begin
       writeln('Неверно');
      end;
end.
```

Результат выполнения программы в случае ввода числа 15:

```
Окно вывода
Сколько будет 3x5?
15
Верно
```

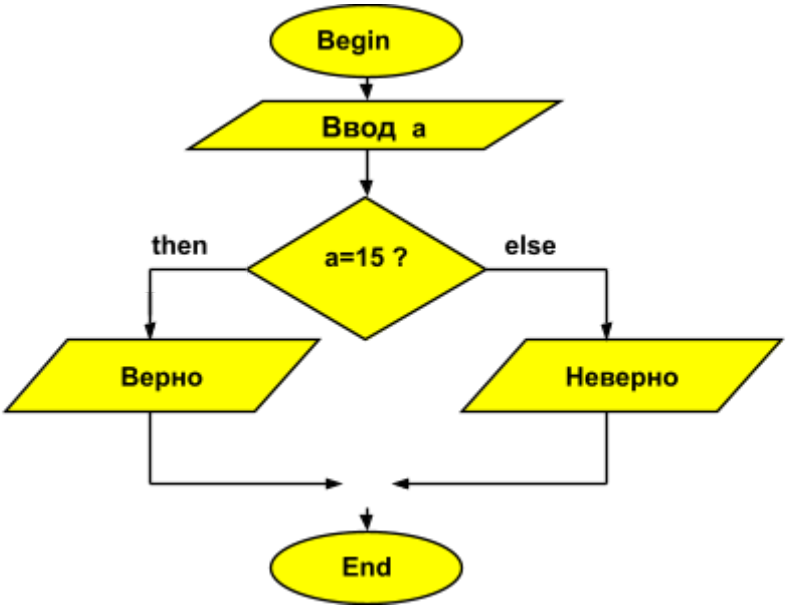
При ином ответе, например, 25, появится сообщение:

```
Окно вывода
Сколько будет 3x5?
25
Неверно
```

Условие – это выражение, стоящее в операторе if..then сразу после слова if. В зависимости от этого условия (его истинности или ложности) компьютер выполняет ту либо иную ветвь программы.

Для обозначения оператора if на блок-схемах используются ромбы, называемые *блоками проверки условия.* Алгоритмическая конструкция ветвления представлена на рисунке.

А вот как будет выглядеть блок-схема алгоритма, реализованного в примере 1.



Простое условие - это математическое сравнение двух выражений по величине (сравнение двух величин).

Операции сравнения на языке программирования можно записать при помощи следующих знаков

Операции сравнения

Знак	***Операция сравнения***
=	Равно
<	Меньше
<=	Меньше либо равно
>	Больше
>=	Больше либо равно
<>	Не равно

Примеры простых условий:

a<>b

A<=0

A+3*c>=20

В качестве оператора_1 и оператора_2 может быть любая из уже известных вам команд.

Рассмотрим еще одну задачу.

Задача 2. С клавиатуры вводятся 2 числа. Вывести на экран большее из них.

```
Program z2;
  var a,b:integer;
begin
  writeln ('Введите числа ');
  readln (a,b);
   if a>b
      then begin
```

```
        write (a);
      end
      else begin
        write (b);
      end;
end.
```

```
Program z2;
   var a,b:integer;
begin
   writeln ('Введите числа ');
   readln (a,b);
   if a>b
    then begin
     write (a);
    end
    else begin
     write (b);
    end;
end.
```

```
Окно вывода
Введите числа
25 54
54
```

Составные условия

При решении различных задач иногда возникает необходимость проверять выполнение двух и более условий. Такие условия называют *составными* (как например, 0<a<5).

Для записи составных условий на языке программирования используют следующие логические операции:

- **and** – логическое «и»;
- **or** – логическое «или»;
- **xor** – логическое «исключающее или»;
- **not** – логическое отрицание.

Простые условия при этом обязательно заключаются в скобки, так как логические операции имеют более высокий приоритет, чем операции сравнения.

Примечания.

Правила выполнения логических операций

1. Составное условие, состоящее из двух простых условий, соединенных операцией and, верно (истинно) только тогда, когда верны оба простых условия.
2. Составное условие, состоящее из двух простых условий, соединенных операцией or, верно тогда, когда верно хотя бы одно из простых условий.
3. Составное условие not верно только тогда, когда простое *условие* ложно.
4. Составное условие, состоящее из двух простых условий, соединенных операцией xor, верно тогда, когда верно только одно из условий.

Задача 3. *Определите, принадлежит ли введенное с клавиатуры число промежутку* [20;140).

```
Program z4;
  var a:integer;
begin
  writeln ('Введите число: ');
  readln (a);
   if (a>=20) and (a<140)
       then begin
        writeln ('Принадлежит');
       end
       else begin
        writeln ('Не принадлежит');
       end;
end.
```

```
Program z4;
   var a:integer;
begin
   writeln ('Введите число: ');
   readln (a);
   if (a>=20) and (a<140)
    then begin
     writeln ('Принадлежит');
    end
    else begin
     writeln ('Не принадлежит');
    end;
end.
```

Окно вывода

```
Введите число:
256
Не принадлежит
```

Сокращенная форма условного оператора

Форма условного оператора

```
If <условия>
   then begin
      <набор операторов 1> ;
   end
   else begin
      <набор операторов 2> ;
   end;
```

называется *полной*.

В языке программирования Pascal существует также *сокращённая форма* условного оператора, которая применяется *в тех случаях, когда какое-либо действие (группу действий) нужно выполнить только при выполнении заданного условия*.

Сокращенная форма условного оператора имеет вид:

```
If <условия>
   then begin
      <набор операторов 1> ;
   end;
```

Задача 4. *С клавиатуры вводится число. Если оно нечетное, уменьшить его на 10.*

```
program z4;
  var a:integer;
begin
 write('Введите целое число: ');
 readln(a);
 if (a mod 2<>0)
   then begin
      a:=a - 10;
   end;
 write(a);
```

end.

```
program z4;
    var a:integer;
begin
  write('Введите целое число: ');
  readln(a);
  if (a mod 2<>0)
   then begin
    a:=a - 10;
   end;
  write(a);
 end.
```

Окно вывода

```
Введите целое число: 48
48
```

Окно вывода

```
Введите целое число: 49
39
```

Составной оператор

При составлении программ на языке программирования часто бывает так, что в случае выполнения либо невыполнения некоторого условия в операторе if необходимо осуществить несколько действий. В этом случае *последовательность действий (несколько операторов подряд) объединяют в одну группу, заключенную между словами begin и end*.

Пример:

```
If x>0 then
        Begin
        x:=x*2;
        write (x);
       end;
```

Такая группа называется ***составным*** **оператором** и рассматривается как единое целое.

Зарезервированные слова **Begin** и **End** часто называют открывающей и закрывающей *операторными скобками.*

Рассмотрим пример, демонстрирующий использование составного оператора.

Задача 5. *С клавиатуры вводятся 2 числа. Если эти числа положительны, найти разность и частное этих чисел, в противном случае – сумму и произведение.*

```
program z5;
   var a,b,c,d:real;
begin
 writeln('Введите числа: ');
 readln(a,b);
 if (a>0) and (b>0)
     then begin
          c:=a-b;
          d:=a/b;
    end
    else begin
          c:=a+b;
          d:=a*b;
   end;
 writeln(c);
 writeln(d);
end.
```

```
program z5;
    var a,b,c,d:real;
begin
  writeln('Введите числа: ');
  readln(a,b);
  if (a>0) and (b>0)
    then begin
      c:=a-b;
      d:=a/b;
     end
     else begin
      c:=a+b;
      d:=a*b;
     end;
  writeln(c);
  writeln(d);
end.
```

Окно вывода

```
Введите числа:
20 12
8
1.66666666666667
```

Алгоритмическая конструкция «Повторение»

Цикл

Нетрудно заметить, что программы, которые составлялись в предыдущих параграфах, обладают одним общим свойством: при их выполнении каждое действие (группа действий) совершается один раз (в линейных программах) либо вообще не совершается (в программах с ветвлением). В жизни часто встречаются случаи, в которых требуется один и тот же набор действий выполнять много раз подряд: «закручивать гайку, пока она не завернется до отказа», «идти, пока не дойдешь…» и т.д.

В программах на языке программирования очень часто необходимо повторять определенные действия. Для этого используется новая форма организации действий – цикл (повторение).

Цикл является одной из базовых алгоритмических конструкций и *представляет собой последовательность действий, которая выполняется неоднократно, до тех пор пока выполняется некоторое условие.* Саму последовательность повторяющихся действий называют **телом цикла**.

Циклы позволяют записать действия в компактной форме. Например, для *вычисления суммы первых десяти натуральных чисел* можно выбрать простое решение и записать вычисление в строчку, употребив 9 операций сложения:

```
Sum:=1+2+3+4+5+6+7+8+9+10;
```

Рассмотрим подробнее *алгоритм вычисления суммы первых 10 натуральных чисел по шагам*. В переменную Sum будем накапливать сумму чисел, в переменную i будем записывать очередное натуральное число.

0 шаг: $sum_0 = 0$
1 шаг: i=1 $sum_1=sum_0+i=0+1=1$
2 шаг: i=2 (i=i+1=1+1) $sum_2=sum_1+i=1+2=3$
3 шаг: i=3 (i=i+1=2+1) $sum_3=sum_2+i=3+3=6$
4 шаг: i=4 (i=i+1=3+1) $sum_4=sum_3+i=6+4=10$
5 шаг: i=5 (i=i+1=4+1) $sum_5=sum_4+i=10+5=15$
. . .

Нетрудно заметить, что при вычислении суммы повторяются всего две операции, причем в определенном порядке:

1. Прибавить слагаемое к ранее полученной сумме:

2. Увеличить на 1 значение слагаемого.

Следовательно, задачу можно решить с помощью следующего алгоритма:

1. Присвоить переменной Sum значение, равное 0 (Sum:=0).
2. Присвоить переменной i (слагаемое) значение, равное 1 (i:=1).
3. Добавить к сумме значение слагаемого i (Sum:= sum+i).
4. Увеличить i на 1 (i:=i+1).
5. Повторить шаги 3 и 4.

Повторив операции 3 и 4 5 либо 9 либо 999 (или более) раз, получим требуемую сумму. Это пример алгоритмической конструкции «Цикл (повторение)».

В языке программирования Pascal имеются разновидности цикла, например:

* *цикл «для»* (for..to/downto);
* *цикл «пока»* (while).

Каждая из этих разновидностей цикла имеет свои особенности.

Оператор цикла с параметром for

Если число повторений цикла известно заранее, то используется алгоритмическая конструкция, которая называется *циклом с параметром (заданным числом повторений)*. На языке Паскаль повторение некоторой последовательности действий известное число раз выполняет оператор цикла **for**.

В общем виде оператор for может быть представлен в двух формах.

Первая форма

```
For i:=N1 to N2 do
begin
  <тело цикла>;
end;
```

Пояснения:

i – *счетчик цикла;*
N1 – *начальное значение;*
N2 – *конечное значение;*
N1 ≤N2.

При использовании данной формы оператора for после каждого выполнения команд тела цикла счетчик цикла автоматически увеличивается на 1.

Вторая форма

```
For i:=N2 downto N1 do
begin
```

<тело цикла>;
end;

Пояснения:

i – *счетчик цикла;*

N2 – *начальное значение;*

N1 – *конечное значение;*

N2 ≥N1.

При использовании данной формы оператора for после каждого выполнения команд тела цикла счетчик цикла автоматически уменьшается на 1.

Тело цикла может быть простым или составным оператором. *Счетчик цикла*, его начальное и конечное значения должны принадлежать к одному и тому же типу данных *(чаще всего это целочисленный тип integer, но могут быть и другие типы, кроме вещественного).*

Решение задач с использованием оператора for.

Пример 1. Вывести на экран 7 раз название нашего города.

```
program z1;
 var
  i:integer;
begin
 for i:=1 to 7 do
  begin
   writeln('Светлогорск');
  end;
end.
```

```
program z1;
 var
  i:integer;
begin
 for i:=1 to 7 do
  begin
   writeln('Светлогорск');
  end;
end.
```

Окно вывода

```
Светлогорск
Светлогорск
Светлогорск
Светлогорск
Светлогорск
Светлогорск
Светлогорск
```

Пример 2. Найти сумму натуральных чисел от 7 до 100.

```
program z2;
 var
  i,s:integer;
begin
 s:=0;
 for i:=7 to 100 do
  begin
   s:=s+i;
  end;
 writeln(s);
end.
```

```
program z2;
 var
  i,s:integer;
begin
 s:=0;
 for i:=7 to 100 do
  begin
   s:=s+i;
  end;
 writeln(s);
end.
```

Окно вывода

```
5029
```

Пример 3. Вывести на экран 20 первых натуральных четных чисел.

```
program z3;
 var
  i,x:integer;
begin
 x:=2;
 for i:=1 to 20 do
  begin
   write(x,' ');
   x:=x+2;
  end;
end.
```

```
program z3;
 var
  i,x:integer;
begin
 x:=2;
 for i:=1 to 20 do
  begin
   write(x,' ');
   x:=x+2;
  end;
end.
```

Окно вывода

```
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
```

Пример 4. Найти сумму чисел от 1 до n, которые делятся на 3, но не делятся на 2.

```
program z4;
 var
  n,i,s:integer;
begin
 readln(n);
 s:=0;
 for i:=1 to n do
  begin
   if (i mod 3=0) and (i mod
2<>0)
    then begin
      s:=s+i;
     end;
  end;
 writeln(s);
end.
```

```
program z4;
 var
  n,i,s:integer;
begin
 readln(n);
 s:=0;
 for i:=1 to n do
  begin
   if (i mod 3=0) and (i mod 2<>0)
   then begin
     s:=s+i;
   end;
  end;
 writeln(s);
end.
```

Окно вывода

```
100
867
```

Оператор цикла с предусловием While.

Оператор цикла for отлично выполняет свои функции, когда число повторений действий заранее известно. Однако такая ситуация встречается в программировании далеко не всегда. Часто приходится решать задачи, когда число повторений действий в теле цикла неизвестно и определяется в ходе решения задачи. В этом случае применяют ***цикл с условием***. В языке программирования Pascal имеется две разновидности цикла с условием:

- *цикл с предварительным условием* - условие цикла проверяется перед выполнением тела цикла;
- *цикл с последующим условием* - условие цикла проверяется после выполнения тела цикла.

Остановимся на цикле с предварительным условием (кратко, с предусловием).

Цикл с предусловием – это цикл, который повторяется до тех пор, пока условие выполняется (истинно).

Для реализации цикла с предусловием используется оператор While. В общем виде оператор While на языке программирования Паскаль может быть представлен так:

```
While <условия> do
Begin
 <тело цикла>;
End;
```

Значение выражения <*условия*>, записанное после слова While, проверяется перед каждым выполнением оператора (операторов) тела цикла.

Если <*условия*> верны (истинны), выполняется тело цикла, и снова вычисляется и проверяется выражение <*условия*>. Если результат проверки <*условия*> неверен (ложный), происходит выход из цикла.

Рассмотрим, как работает оператор While *на известном нам примере вычисления суммы N первых натуральных чисел*.

Примечания.

1. В цикле While счетчик цикла можно изменять на любое число, даже дробное.
2. Цикл While выполняется до тех пор, пока условия, записанные после слова While, истинны.
3. Выполнение цикла While прекращается, как только условие примет ложное значение.
4. В операторах for и while точка с запятой не ставится ни перед словом do, ни после него!

Пример 2. Найдите сумму натуральных чисел от 1 до 1000, делящихся нацело на 17.

```
program z2;
var
  i,s:integer;
begin
  s:=0;
  i:=17;
while i<=1000 do
  begin
   s:=s+i;
   i:=i+17;
  end;
  writeln(s);
end.
```

```
program z2;
   var
      i,s:integer;
begin
  s:=0;
  i:=17;
  while i<=1000 do
   begin
      s:=s+i;
      i:=i+17;
   end;
  writeln(s);
end.
```

Окно вывода

```
29087
```