Основные алгоритмические конструкции языка Pascal

1. линейные;
2. ветвящиеся;
3. циклические.

Виды алгоритмов

В линейном алгоритме операции выполняются последовательно, в порядке их записи.

Каждая операция является самостоятельной, независимой от каких-либо условий.

На схеме блоки, отображающие эти операции, располагаются в линейной последовательности.

Линейные алгоритмы

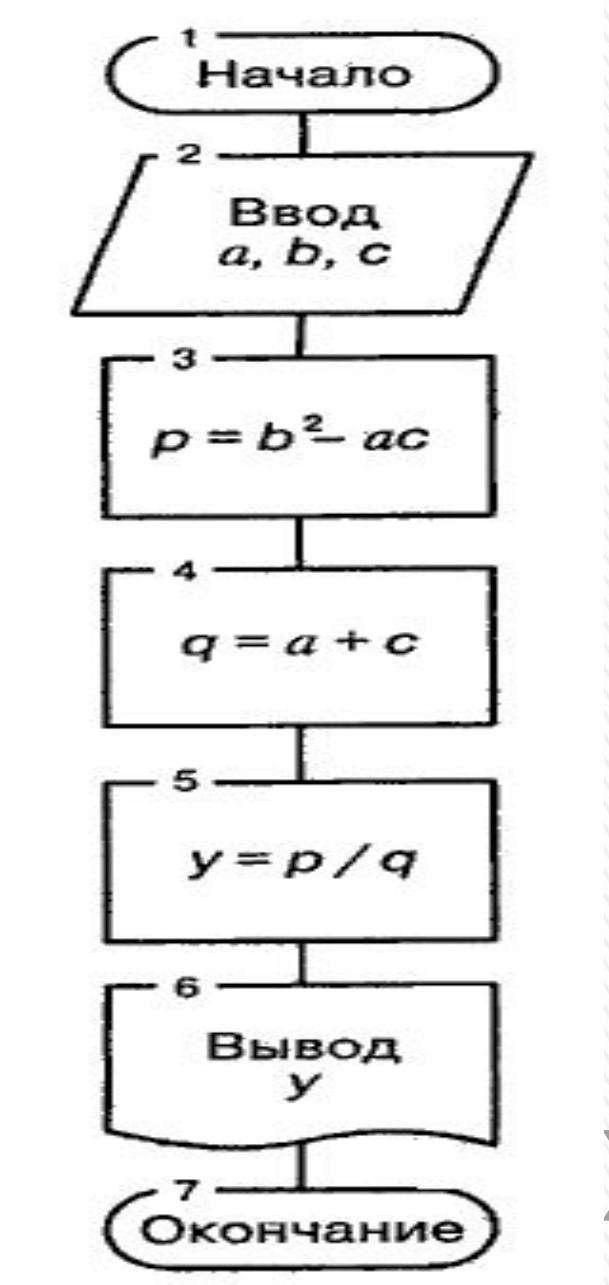
Линейные алгоритмы имеют место, например, при вычислении арифметических выражений, когда имеются конкретные числовые данные и над ними выполняются соответствующие условию задачи действия.

Линейные алгоритмы

Пример линейного алгоритма

Составить блок – схему алгоритма вычисления арифметического выражения

y=(b2-ac):(a+c)



Разветвляющиеся алгоритмы

Алгоритм называется ветвящимся, если для его реализации предусмотрено несколько направлений (ветвей).

Каждое отдельное направление алгоритма обработки данных является отдельной ветвью вычислений.

Алгоритм с ветвлением

Ветвление в программе — это выбор одной из нескольких последовательностей команд при выполнении программы.

Выбор направления зависит от заранее определенного признака, который может относиться к исходным данным, к промежуточным или конечным результатам.

Признак характеризует свойство данных и имеет два или более значений.

Алгоритмы с ветвлением

Ветвящийся процесс, включающий в себя две ветви, называется простым, более двух ветвей — сложным.
Сложный ветвящийся процесс можно представить с помощью простых ветвящихся процессов.

Алгоритм с ветвлением

Направление ветвления выбирается логической проверкой, в результате которой возможны два ответа:

1. «да» — условие выполнено
2. «нет» — условие не выполнено.

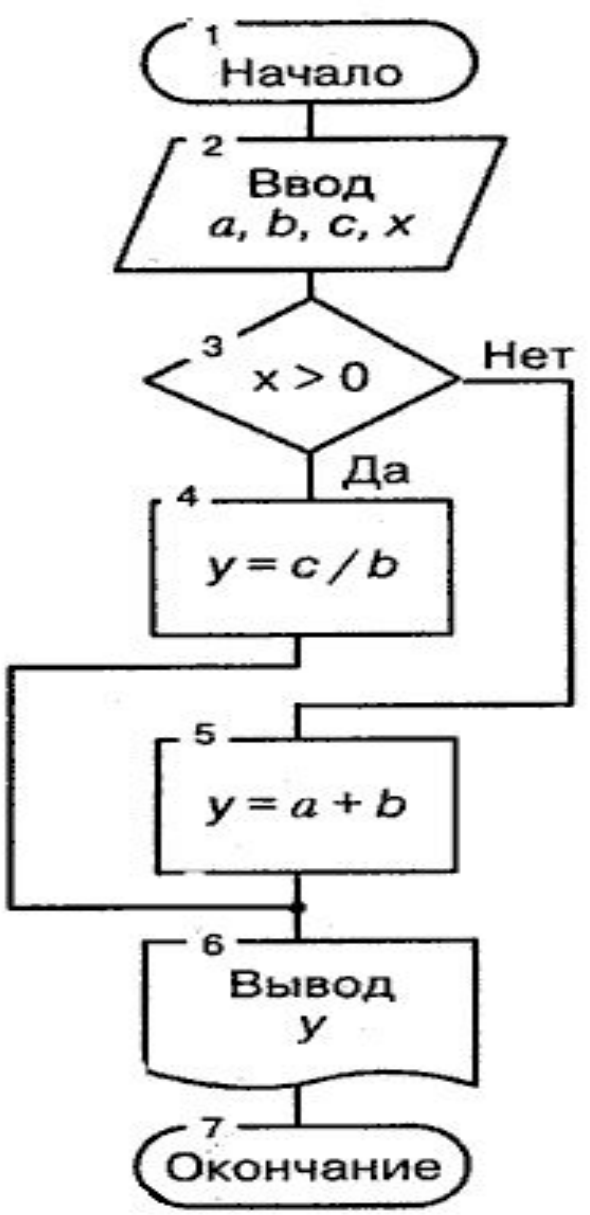
Алгоритм с ветвлением

Алгоритм с полным ветвлением

Пример алгоритма с ветвлением

Составить блок-схему алгоритма с ветвлением для вычисления следующего выражения:

Y = (a+b), если X <0;
c/b, если X>0.



if <условие>

then <команда, выполняемая при выполнении условия>

else <команда, выполняемая при невыполнении условия>;

Кодирование ветвления в полной форме

Ключевые (служебные) слова Паскаля
– if (если), then (то), else (иначе).

Кодирование ветвления в полной форме

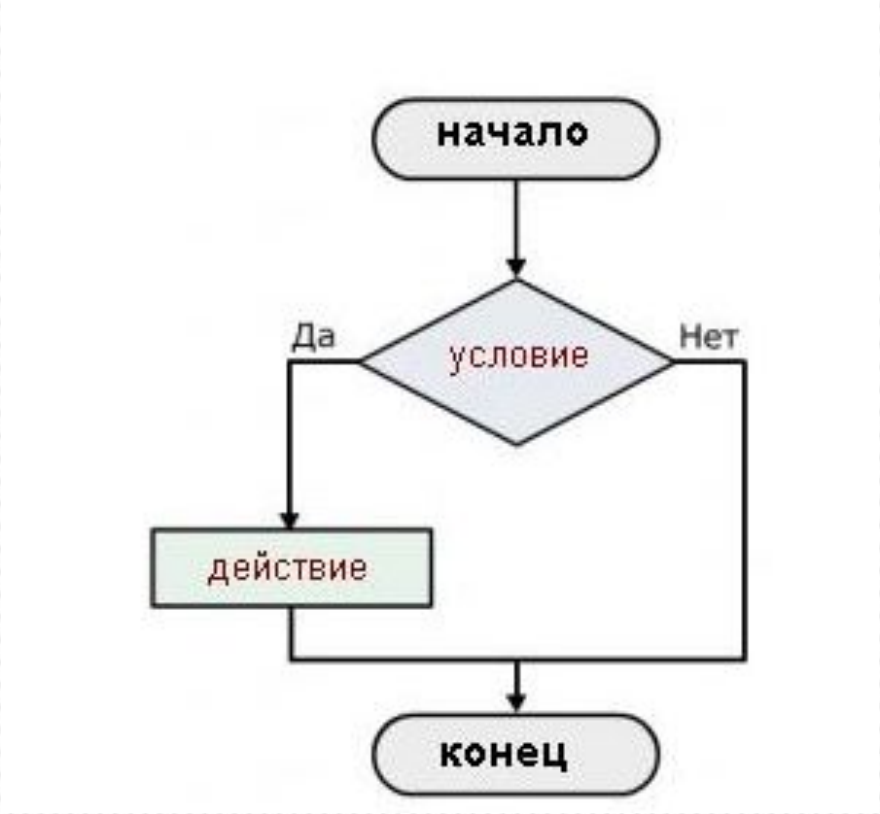
Пример.

if (x>y) { если текущее значение x больше текущего значения y, }

then y := x { то текущее значение y полагаем равным текущему значению x, }

else x:= y; { иначе (при x <= y) текущее значение x заменяем на текущее значение y }.

Кодирование ветвления в неполной форме



Алгоритм с неполным ветвлением

if <условие>

then <команда, выполняемая при выполнении условия>;

Кодирование ветвления в неполной форме

Простой оператор не содержит в себе других операторов (оператор присваивания, вызов процедуры,…).

Два последовательных оператора должны разделяться точкой с запятой (имеет смысл конца оператора):

```
a := 11; b := a * a; Write(a,b);
```

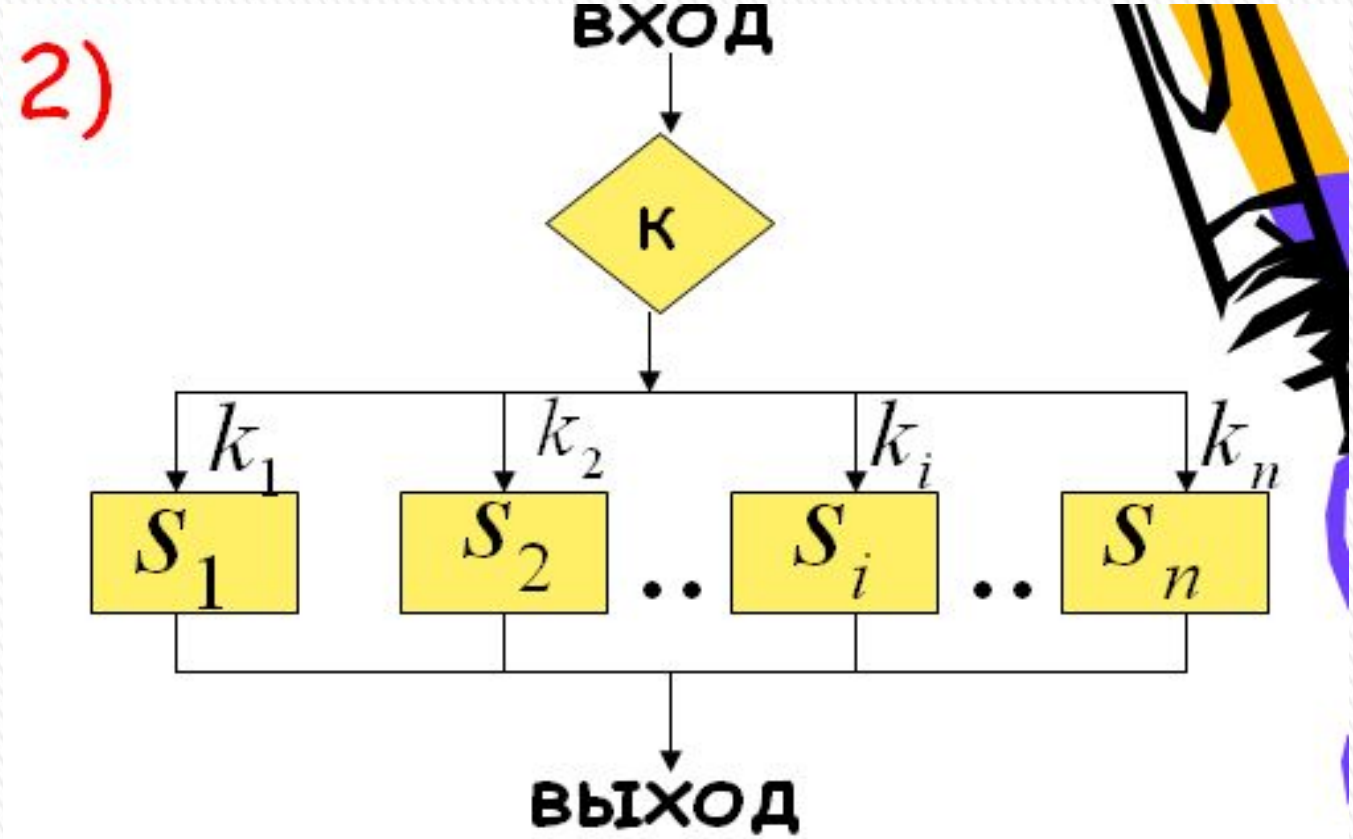
Простой и составной операторы

Составной оператор – это последовательность операторов, рассматриваемых как единый. Оформляется с помощью зарезервированных слов begin и end (операторные скобки).

Простой и составной операторы

```
begin
        a := 11;
        b := a * a;
        Write(a,b)
    end;
```

Простой и составной операторы



Алгоритм выбора

Алгоритмическая структура «выбор» применяется для реализации ветвлений со многими вариантами серий команд.

В структуру выбора входят несколько условий, которые последовательно проверяются.

Команда выбора

При истинности одного из условий Условие 1, Условие 2 и т. д. выполняется соответствующая последовательность команд Серия 1, Серия 2 и т. д.

Если ни одно из условий не истинно, то выполняется последовательность команд Серия.

Команда выбора

```
3) Case <выражение> of
 значение 1:
выполняемый оператор 1;
 значение 2:
выполняемый оператор 2;
 --------------
 значение n:
выполняемый оператор n;
end;
```

Команда выбора

```
case I of
      1    :  X := X +1;
      2,3  :  X := X +2;
      4..9 :  begin
                Write(X);
                X := X + 3
              end
    else
      X := X * X;
     Writeln(X)
end;
```

Команда выбора

Циклические алгоритмы

Циклическими называются алгоритмы, содержащие циклы.

Цикл — это многократно повторяемый участок алгоритма.

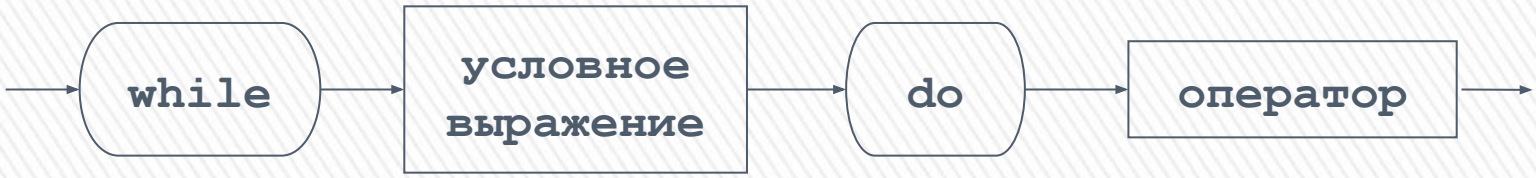
Циклические алгоритмы

Виды циклов

Цикл называется детерминированным, если число повторений тела цикла заранее известно или определено.

Цикл называется итерационным, если число повторений тела цикла заранее неизвестно, а зависит от значений параметров (некоторых переменных), участвующих в вычислениях.

Оператор цикла "Пока" (с предусловием)



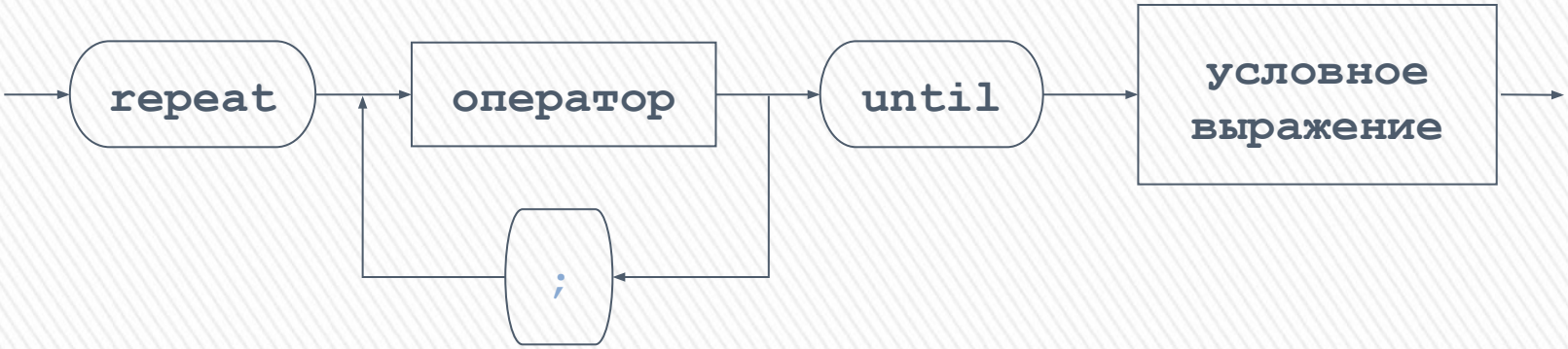
Оператор цикла "Пока" (с предусловием)

<Оператор> (тело цикла), стоящий после служебного слова do, будет выполняться циклически до тех пор, пока выполняется логическое условие, т.е. пока значение <условного выражения> равно True.

Оператор цикла "Пока" (с предусловием)

```
Var    F,N : LongInt;        {вычисление 10!}
Begin
  F := 1; N := 1;
  while N <= 10 do
    begin
      F := F * N; Inc(N)     {N := N + 1}
    end;
  Writeln(F)
End.
```

Оператор цикла "До" (с постусловием)



Оператор цикла "До" (с постусловием)

Операторы между словами repeat и until образуют тело цикла.

Если <условное выражение> имеет значение True, то цикл завершается.

Оператор цикла "До" (с постусловием)

repeat
<тело цикла>
{ операторы begin ... end не требуются! }
until <логическое условие>;

Оператор цикла "До" (с постусловием)

Использование оператора repeat ... until оправдано тогда, когда нужны повторяющиеся действия, от выполнения которых зависит дальнейшее продолжение цикла.

Оператор цикла "До" (с постусловием)

Цикл "Пока" - "пока условие истинно, выполнять операторы тела".
Цикл "До" - "выполнять тело цикла до тех пор, пока не станет истинным условие";

Оператор цикла с параметром (цикл по счетчику)

Используется для организации "строгих" циклов, которые должны быть проделаны заданное число раз.

Оператор цикла с параметром (цикл по счетчику)

<Параметр цикла> – переменная порядкового типа, к этому же типу должны относиться значения <выражения 1> и <выражения 2>.

Оператор цикла с параметром (цикл по счетчику)

Значение <параметра цикла> меняется в возрастающем (при использовании зарезервированного слова to) или убывающем (downto) порядке от значения <выражения 1> до значения <выражения 2> с постоянным шагом, равным интервалу между двумя ближайшими значениями в типе, к которому относится <параметр цикла> (для целочисленных типов - это 1, для символьного - от одного символа к другому при увеличении кода на 1, и т.д.).

Оператор цикла с параметром (цикл по счетчику)

```
for <счетчик1> := <значение1> to
<конечное_значение> do <оператор1>;
```

Оператор цикла с параметром (цикл по счетчику)

```
for <счетчик2> := <значение2> downto
<конечное_значение> do <оператор1>;
```