Информатика и ИКТ

# НАЧАЛА ПРОГРАММИРОВАНИЯ


# ОБЩИЕ СВЕДЕНИЯ О ЯЗЫКЕ ПРОГРАММИРОВАНИЯ ПАСКАЛЬ

9 класс

Издательство БИНОМ

# СОДЕРЖАНИЕ ПРЕЗЕНТАЦИИ:

***Языки программирования*** - это формальные языки, предназначенные для записи алгоритмов, исполнителем которых будет компьютер.

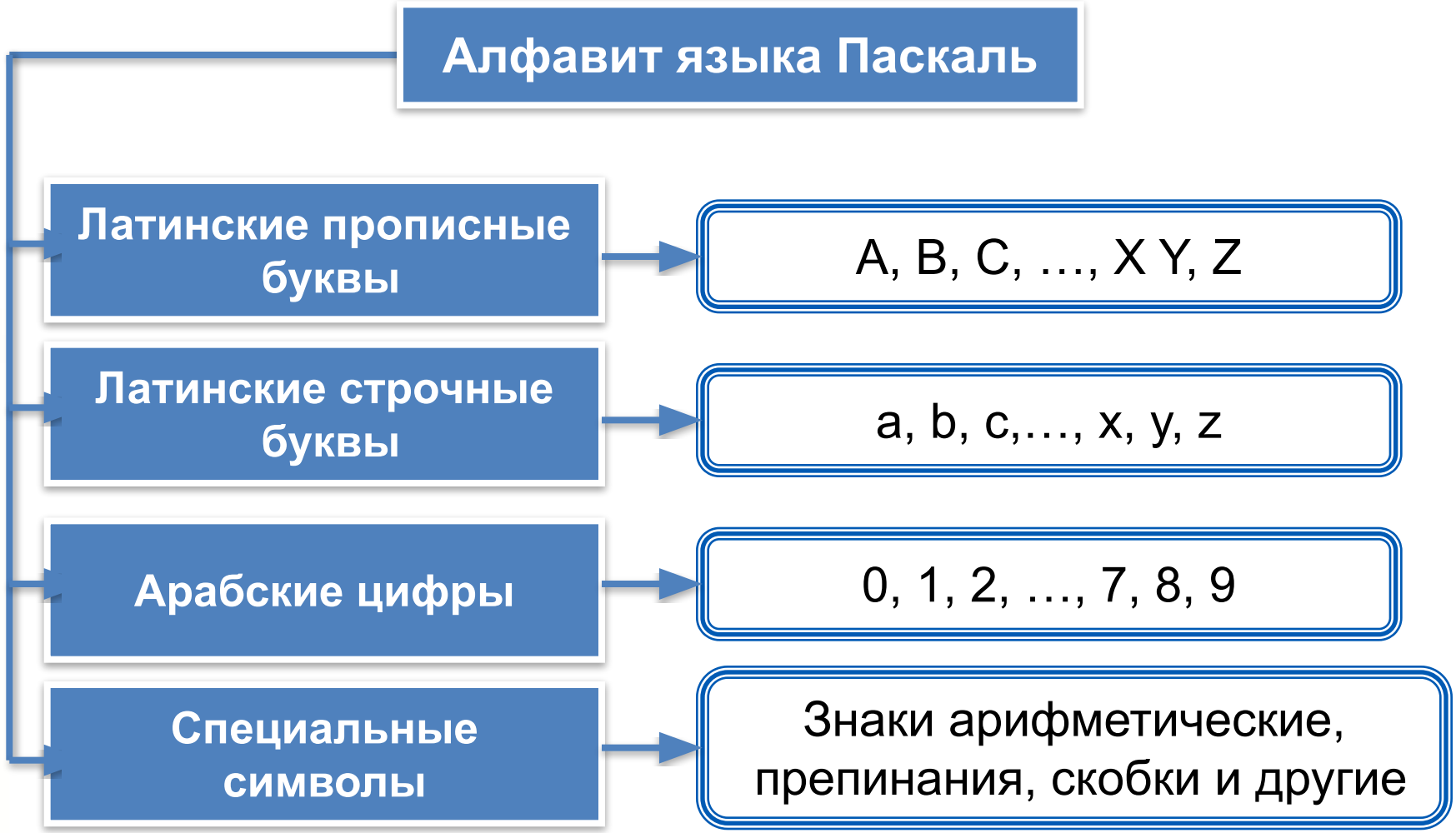
Записи алгоритмов на языках программирования называются ***программами***.

***Язык Паскаль*** – универсальный язык программирования.

**Никлаус Вирт** (род. в 1934 г.) - швейцарский учёный, специалист в области информатики, один из известнейших теоретиков в области разработки языков программирования, профессор информатики (компьютерных наук). Разработчик языка Паскаль и ряда других языков программирования.

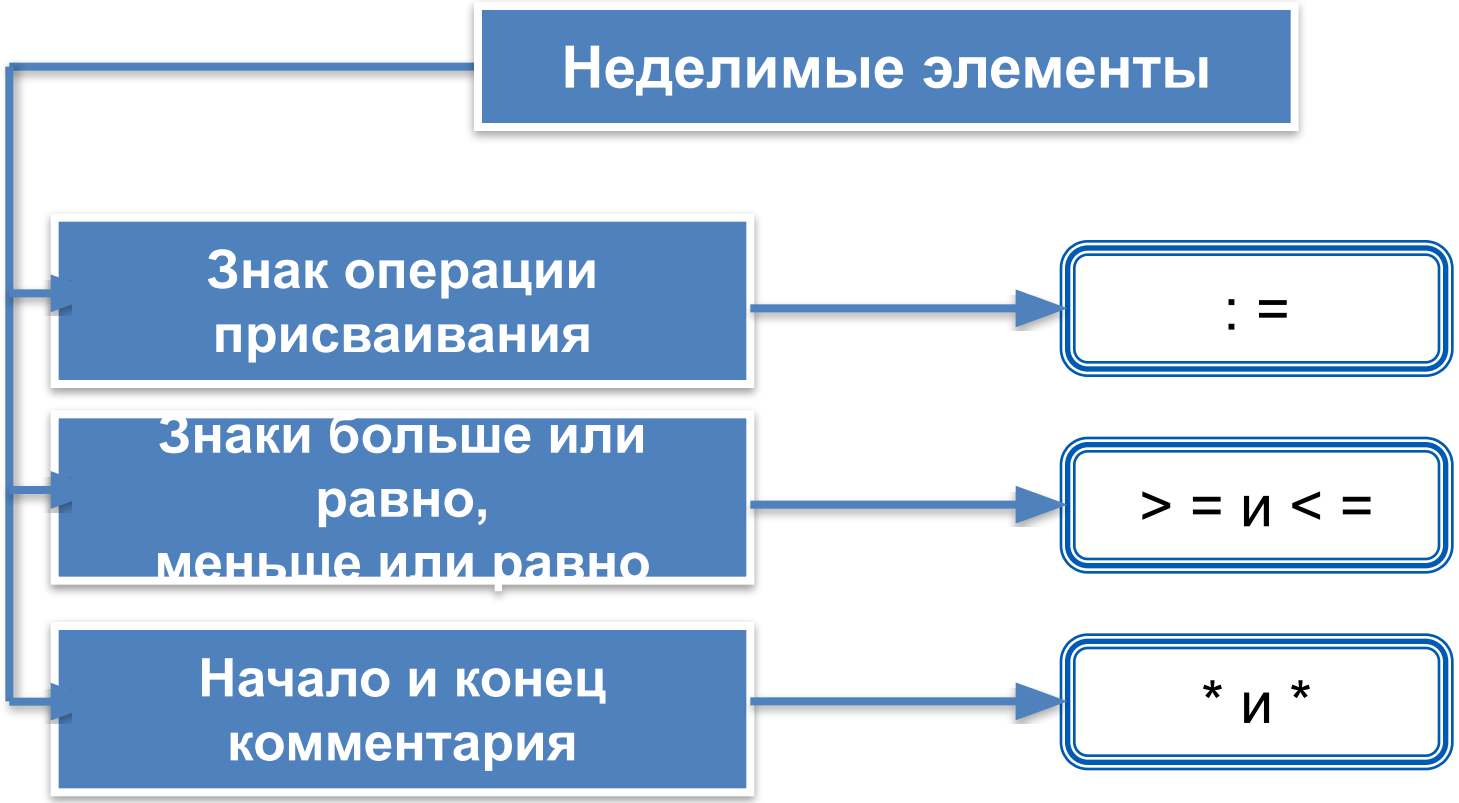
# Алфавит языка

Алфавит языка программирования Паскаль - набор допустимых символов, которые можно использовать для записи программы.



# Алфавит языка

В алфавит языка Паскаль включены неделимые элементы (составные символы).



# Словарь языка

| Служебное слово языка Паскаль | Значение служебного слова |
|---|---|
| **and** | и |
| **array** | массив |
| **begin** | начало |
| **do** | выполнить |
| **else** | иначе |
| **for** | для |
| **if** | если |
| **of** | из |
| **or** | или |
| **procedure** | процедура |
| **program** | программа |
| **repeat** | повторять |
| **then** | то |
| **to** | до (увеличивая до) |
| **until** | до (до тех пор, пока) |
| **var** | переменная |
| **while** | пока |

# Алфавит и словарь языка

***Имена*** (констант, переменных, программ и других объектов) - любые отличные от служебных слов последовательности букв, цифр и символа подчеркивания, начинающиеся с буквы или символа подчеркивания.

**Правильные имена**

**x**
**velichina**
**zzz**
**polnaja_summa**
**tri_plus_dva**
**s25**
**_k1**
**a1b88qq**
**oshibka**

**Неправильные имена**

**Ж** - буква не латинского алфавита
**polnaja summa** - содержится символ (пробел), не являющийся буквой, цифрой или знаком подчеркивания.
**2as** - начинается с цифры
**Domby&Son** - содержится символ &, не являющийся буквой, цифрой или знаком подчеркивания

Прописные и строчные буквы в именах не различаются. Длина имени может быть любой.

# Простые типы данных

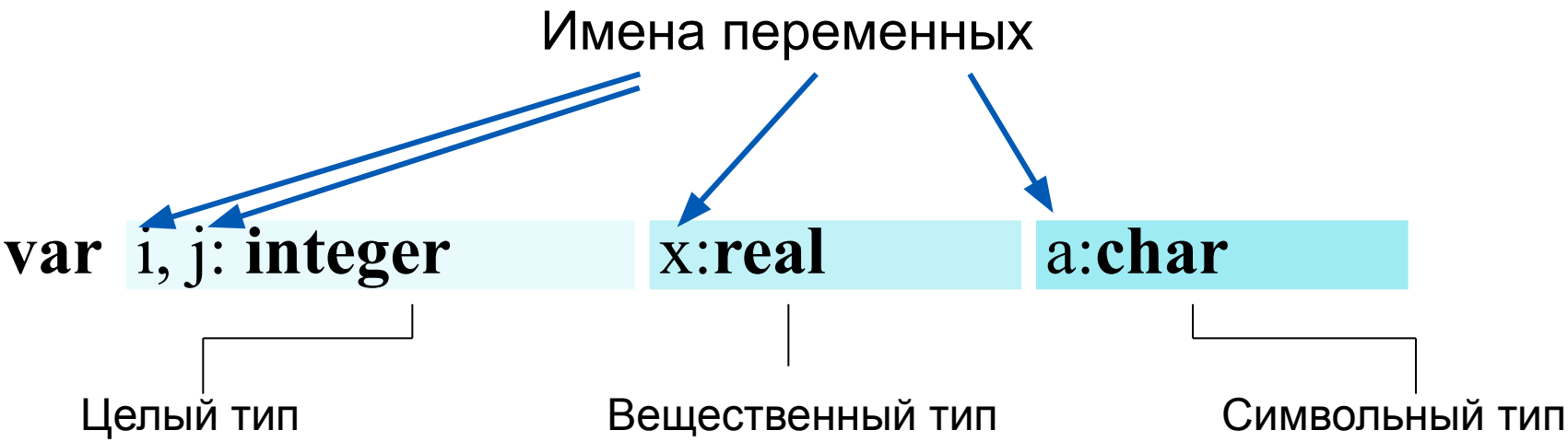
| Название | Обозначение | Допустимые значения | Область памяти |
|---|---|---|---|
| Целочисленный | integer | - 32 768… 32 768 | 2 байта со знаком |
| Вещественный | real | $\pm(2.9 * 10^{-39} \ldots 1.7 * 10^{+38})$ | 6 байтов |
| Символьный | char | Произвольный символ алфавита | 1 байт |
| Строковый | string | Последовательность символов длиной меньше 255 | 1 байт на символ |
| логический | boolean | True и False | 1 байт |

Типы величин

Ôàéë "SWF"

# Структура программы на языке Паскаль

# Раздел описания переменных



# Общий вид программы

```
program <имя программы>;
  const <список постоянных значений>;
  var <описание используемых переменных>;
begin <начало программного блока>
   <оператор 1>;
   <оператор 2>;
       …
   <оператор n>
end.
```

**Операторы** - языковые конструкции для записи действия, выполняемого над данными в процессе решения задачи.

# Оператор присваивания

Основное преобразование данных, выполняемое компьютером, - присваивание переменной нового значения, что означает изменение содержимого области памяти.

Общий вид оператора:

**<имя переменной>:=<выражение>**

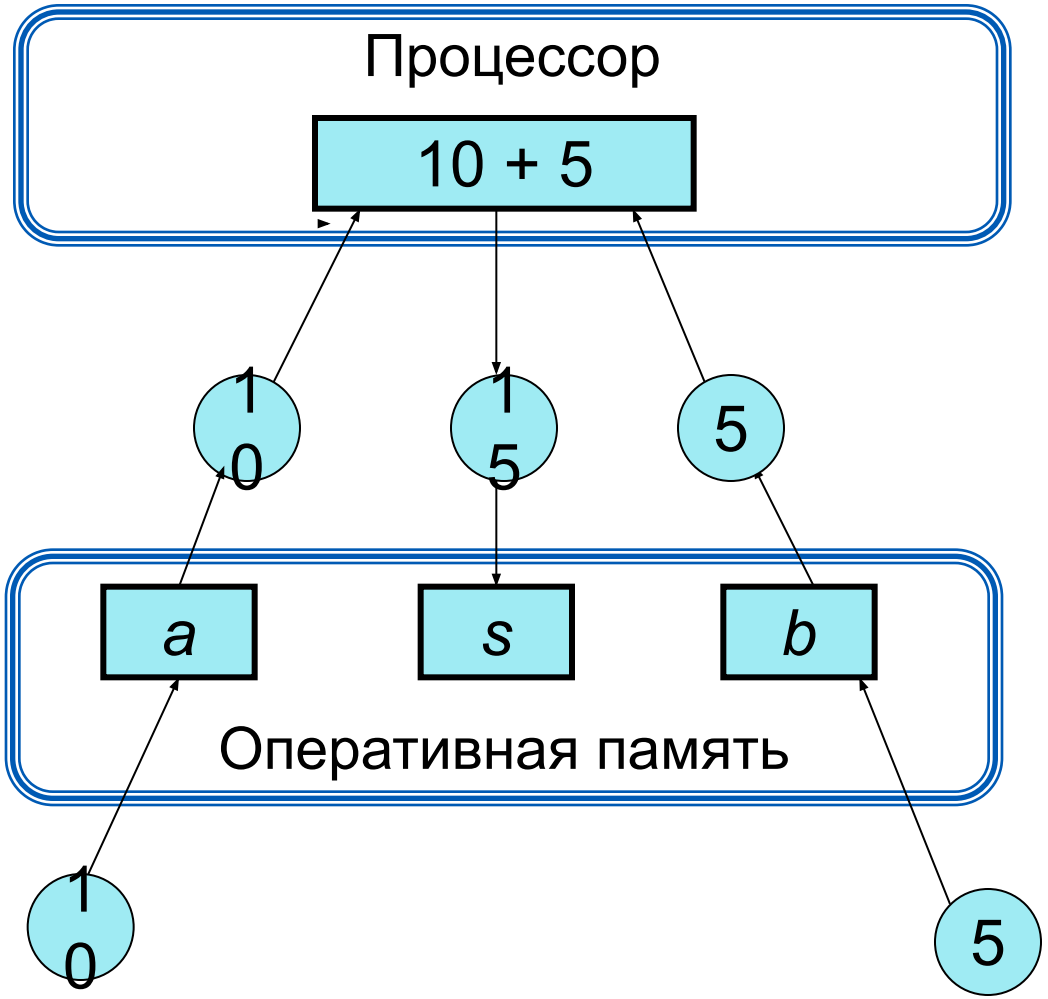
```
a:=10;
b:=5;
s:=a+b;
```

**Команда присваивания**

Ôàéë "SWF"

# Выполнение оператора присваивания

a:=10;
b:=5;
s:=a+b



# НАЧАЛА ПРОГРАММИРОВАНИЯ


## Организация ввода и вывода данных.

- **оператор вывода** writer
- **формат вывода**
- **оператор ввода** read

9 класс

Издательство БИНОМ

# Вывод данных

Вывод данных из оперативной памяти на экран монитора:

**write** (<выражение 1> ,< выражение 2> , ...,< выражение N>)

список вывода

**Выражения** - символьные, числовые, логические, в том числе переменные и константы

***Пример:***

**write** ('s=', s). Для s=15 на экране будет: s=15.

**Информация в кавычках выводится на экран без изменений**

# Варианты организации вывода

| Вариант организации вывода | Оператор вывода | Результат |
|---|---|---|
| Без разделителей | **write** (1, 20, 300). | 120300 |
| Добавить разделители – запятые | **write** (1, ’,’ , 20, ’, ’, 300) | 1, 20, 300 |
| Добавить разделители – пробелы | **write** (1, ‘ ‘, 2, ‘ ‘, 3) | 1 20 300 |

# Формат вывода

**Формат вывода** позволяет установить количество позиций на экране, занимаемых выводимой величиной.

**write** (s:x:y)

**x** - общее количество позиций, отводимых под число;

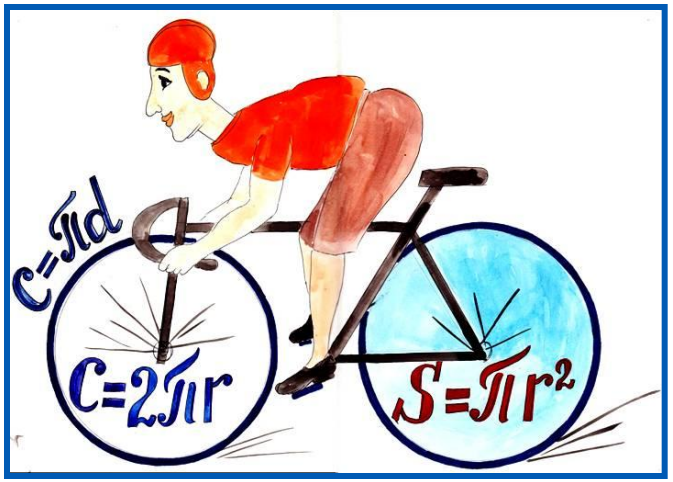
**y** - количество позиций в дробной части числа.

| Оператор вывода | Результат выполнения оператора |
|---|---|
| **write** (‘s=‘, s:2:0); | s=15 |
| **write** (‘s=‘, s:3:1); | s=15.0 |
| **write** (‘s=‘, s:5:1); | s= 15.0 |

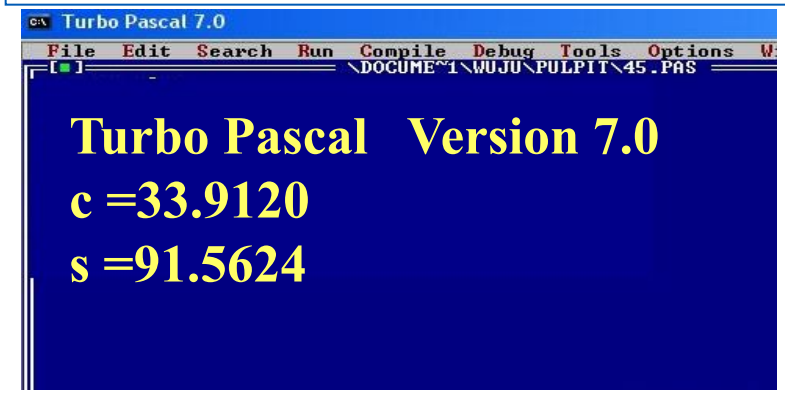
**writeln** - вывод с новой строки!

# Первая программа

```
program n_1;
   const pi=3.14;
   var r, c, s: real;
begin
  r:=5.4;
  c :=2*pi*r;
  s:=pi*r*r;
  writeln ('c =', c:6:4);
  writeln ('s=', s:6:4)
end.
```




Результат работы программы:



# Ввод данных с клавиатуры

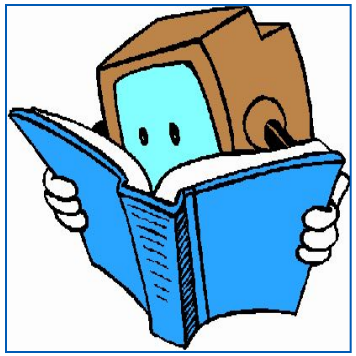
Ввод в оперативную память значений переменных :

**read** (<имя переменной1>, …, <имя переменной N> )

список ввода

Выполнение оператора **read:**

1) компьютер переходит в режим ожидания данных:
2) пользователь вводит данные с клавиатуры:
   - несколько значений переменных числовых типов могут вводиться через пробел или через запятую;
   - при вводе символьных переменных пробел и запятую ставить их нельзя;
3) пользователь нажимает клавишу **Enter**.



# Ввод данных с клавиатуры

! ***Типы вводимых значений*** должны ***соответствовать типам переменных***, указанных в разделе описания переменных.

```
var i, j: integer;x: real;a: char;
read (i, j, x, a);
```

варианты организации входного потока:

1 0 2.5 A<**Enter**>

1,0 <**Enter**>

1<**Enter**>

2.5, A<**Enter**>

0<**Enter**>

2.5<**Enter**>

A<**Enter**>

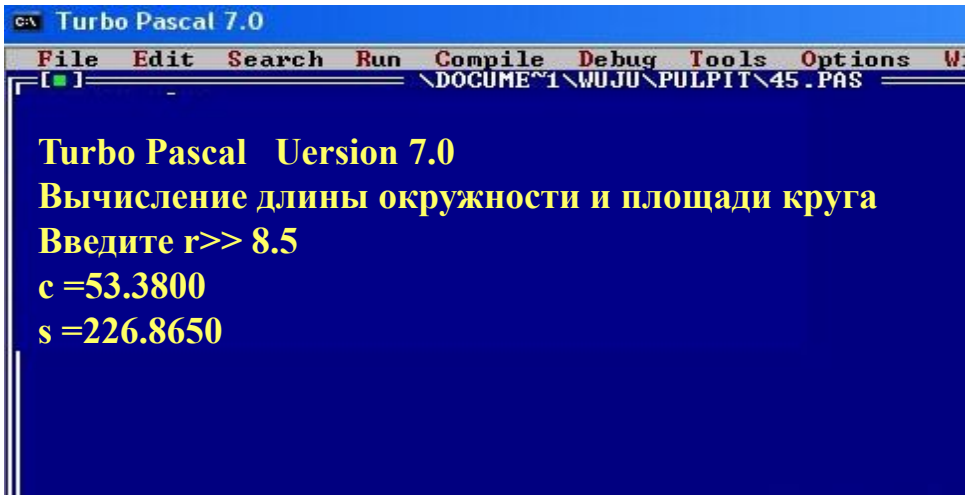
После выполнения оператора **readln** курсор переходит на новую строку.



# Улучшенная программа

```
program n_1;
  const pi=3.14;
  var r, c, s: real;
begin
  writeln('Вычисление длины окружности и площади круга');
  write('Введите r>>');
  readln(r);
  c:=2*pi*r;
  s:=pi*r*r;
  writeln ('c =', c:6:4);
  writeln ('s=', s:6:4)
end.
```

Результат работы программы:



# НАЧАЛА ПРОГРАММИРОВАНИЯ

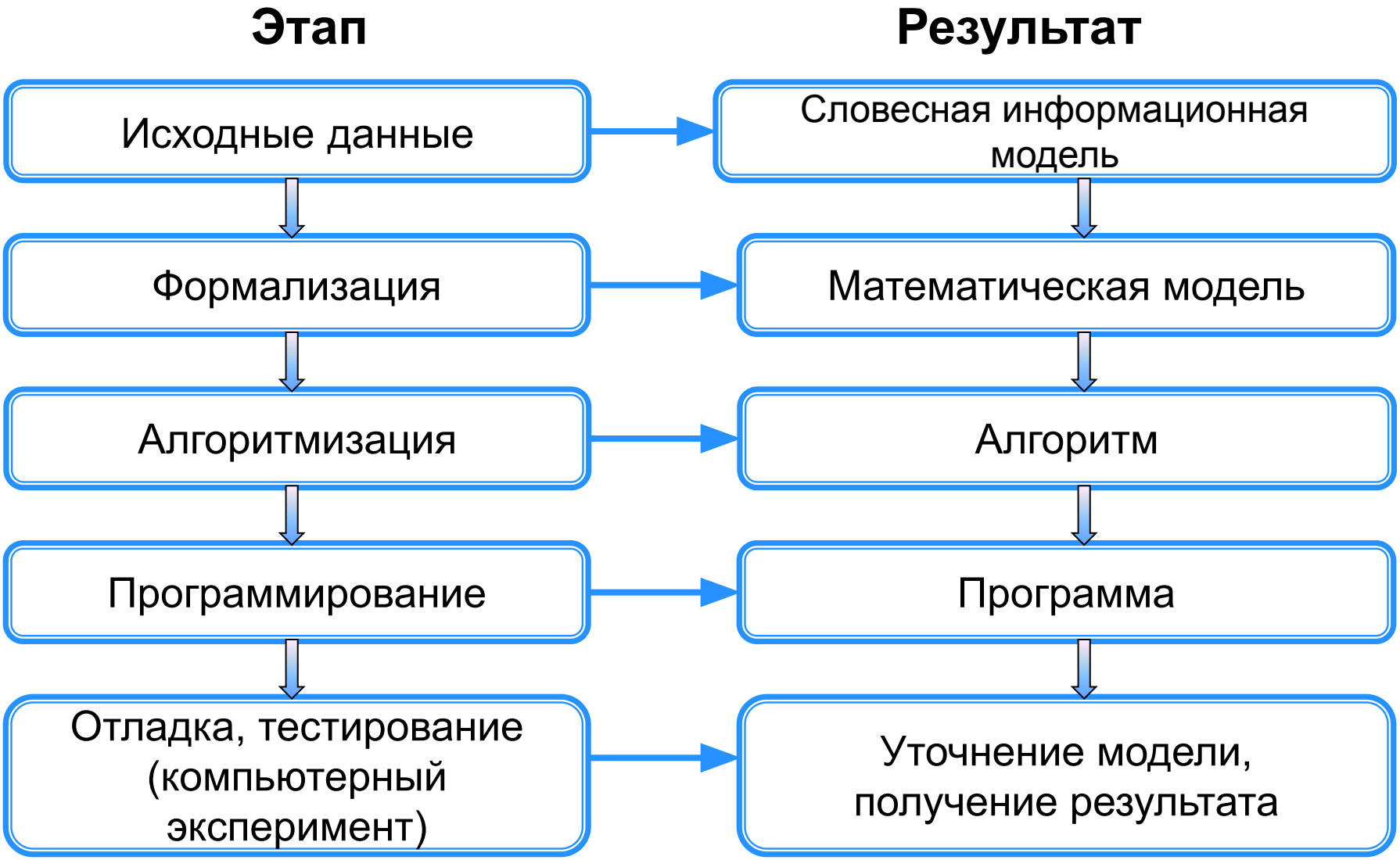

## Программирование как этап решения задач на компьютере

- постановка задачи
- формализация
- алгоритмизация
- программирование
- отладка и тестирование

9 класс

Издательство БИНОМ

# Этапы решения задач на компьютере



# Задача о пути торможения автомобиля

Водитель автомобиля, движущегося с некоторой постоянной скоростью, увидев красный свет светофора, нажал на тормоз. После этого скорость автомобиля стала уменьшаться каждую секунду на 5 метров. Требуется найти расстояние, которое автомобиль пройдёт до полной остановки.

***Первый этап***

Дано:

$v_{0x}$ - начальная скорость;

$v_x$ - конечная скорость (равна нулю);

$a_x$ - ускорение (равно -5 м/с)

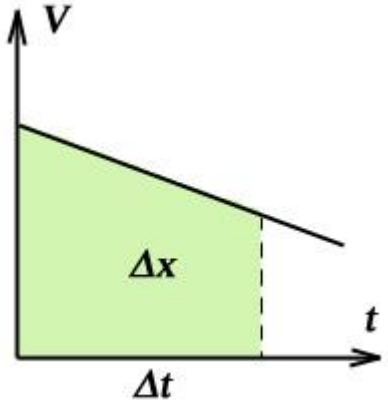
Требуется найти: расстояние, которое пройдет автомобиль до полной остановки.

# Задача о пути торможения автомобиля

***Второй этап***

В данной ситуации мы имеем дело с прямолинейным равноускоренным движением тела. Формула для перемещения при этом имеет вид:

$$s_x = \frac{v_{0x}(v_x - v_{0x})}{a_x} + \frac{a_x}{2}\left(\frac{v_x - v_{0x}}{a_x}\right)^2$$




Упростим эту формулу с учётом [illegible] ная скорость равна нулю:

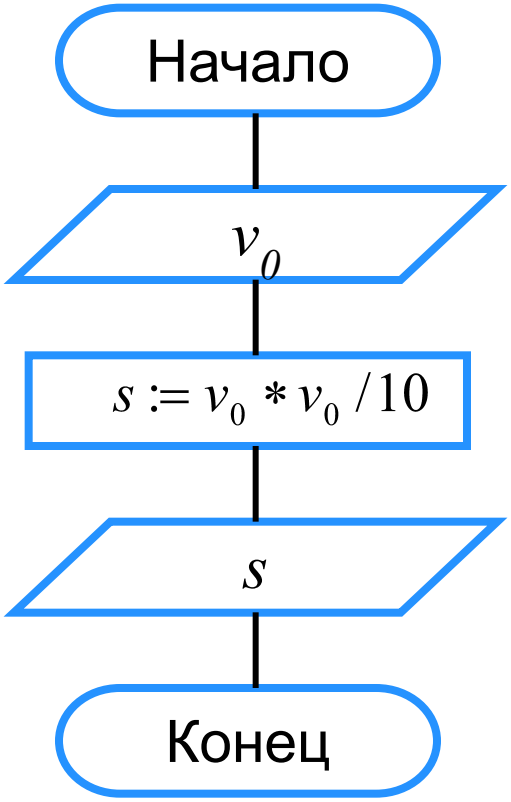
$$s_x = \frac{v_{0x}^2}{2a_x}$$

При $a_x$ = - 5м/с получим: $s_x = \frac{v_{0x}^2}{10}$

# Задача о пути торможения автомобиля

***Третий этап***

Представим алгоритм решения задачи в виде блок-схемы:




# Задача о пути торможения автомобиля

***Четвёртый этап***

Запишем данный алгоритм на языке программирования Паскаль:

```
program n_2;
    var v0, s: real;
begin
   writeln('Вычисление длины пути торможения автомобиля');
   write('Введите начальную скорость (м/с)> ');
   readln (v0);
   s:=v0*v0/10;
   writeln ('До полной остановки автомобиль пройдет', s:8:4,' м.')
end.
```

# Задача о пути торможения автомобиля

***Пятый этап***

Протестировать составленную программу можно, используя ту информацию, что при скорости 72 км/ч с начала торможения до полной остановки автомобиль проходит 40 метров.

Выполнив программу несколько раз при различных исходных данных, можно сделать вывод: чем больше начальная скорость автомобиля, тем большее расстояние он пройдет с начала торможения до полной остановки.

# НАЧАЛА ПРОГРАММИРОВАНИЯ


## Программирование линейных алгоритмов

- вещественный тип данных
- целочисленный тип данных
- символьный тип данных
- строковый тип данных
- логический тип данных

9 класс

Издательство БИНОМ

# Числовые типы данных

Стандартные функции языка Паскаль:

| Функция | Назначение | Тип аргумента | Тип результата |
|---|---|---|---|
| abs (*x*) | Модуль *x* | integer, real | Такой же, как у аргумента |
| sqr (*x*) | Квадрат *x* | integer, real | Такой же, как у аргумента |
| sqrt (*x*) | Квадратный корень из *x* | integer, real | real |
| round (*x*) | Округление *x* до ближайшего целого | real | |
| frac (*x*) | Целая часть *x* | real | |
| int (*x*) | Дробная часть *x* | real | |
| random | Случайное число от 0 до 1 | - | real |
| random (*x*) | Случайное число от 0 до *x* | integer | integer |

# Исследование функций round, int и frac

```
program n_3;
 var x: real;
begin
  writeln ('Исследование функций round, int, frac');
  write ('Введите x>>');
  readln (x);
  writeln ('Округление - ', round(x));
  writeln ('Целая часть - ', int(x));
  writeln ('Дробная часть - ', frac(x))
end.
```

Выполните программу несколько раз для
$x \in$ {10,2; 10,8; –10,2; –10,8}.
Какой будет тип результата каждой из этих функций?

?

# Целочисленный тип данных

Операции над целыми числами в языке Паскаль:

| Операция | Обозначение | Тип результата |
|---|---|---|
| Сложение | + | integer |
| Вычитание | - | integer |
| Умножение | * | integer |
| Получение целого частного | div | integer |
| Получение целого остатка деления | mod | integer |
| Деление | / | real |

# Операции div и mod

Трёхзначное число можно представить в виде следующей суммы: $x = a \cdot 100 + b \cdot 10 + c$, где $a, b, c$ - цифры числа.

Программа нахождения суммы цифр вводимого с клавиатуры целого трёхзначного числа.

```
program n_4;
  var x, a, b, c, s: integer;
begin
  writeln ('Нахождение суммы цифр трёхзначного числа');
  write ('Введите исходное число>>');
  readln (x);
  a:=x div 100;
  b:=x mod 100 div 10;
  c:=x mod 10;
  s:=a+b+c;
  writeln ('s= ', s)
end.
```

# Символьный и строковый типы данных

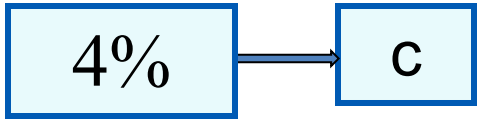
**Символы** – это все буквы и значки, которые есть на клавиатуре. Для ввода в программу символьных переменных необходимо указать для них символьный тип данных **char**.

Функция **ord** преобразовывает букву в её числовой код.

Функция **chr** преобразовывает числовой код символа в сам символ.


Значением строковой величины (тип **string**) является произвольная последовательность символов, заключенная в апострофы.

```
var c: string
c:= chr(52)+chr(37)
```



# Символьный и строковый типы данных

```
program n_5;
  var a: char; kod: integer; b: string;
begin
  writeln ('Код и строка');
  write ('Введите исходную букву>>');
  readln (a);
  kod:=ord(a);
  b:=chr(kod–1)+a+chr(kod+1);
  writeln ('Код буквы ', a, '-', kod);
  writeln ('Строка: ', b)
end.
```

Вывод на экран кода буквы, введённой с клавиатуры

Вывод на экран строки из трёх букв. Каких?

# Логический тип данных

Величины логического типа принимают всего два значения:

**false** и **true;**

**false** < **true**.

Логические значения получаются в результате выполнения операций сравнения числовых, символьных, строковых и логических выражений.

В Паскале логической переменной можно присваивать результат операции сравнения.

# Логический тип данных

Пусть **ans** - логическая переменная,

***n*** - целая переменная.

В результате выполнения оператора присваивания

**ans:=n mod 2=0**

переменной **ans** будет присвоено значение **true** при любом чётном ***n*** и **false** в противном случае.

```
program  n_6;
  var n: integer; ans: boolean;
begin
  writeln ('Определение истинности высказывания о чётности числа');
  write ('Введите исходное число>>');
  readln (n);
  ans:=n mod 2=0;
  writeln ('Число ', n,' является четным - ', ans)
end.
```

# Логический тип данных

Логическим переменным можно присваивать значения логических выражений, построенных с помощью логических функций **и (and), или (or), не (not).**

| Логическая операция в Паскале | Название операции |
|---|---|
| and | конъюнкция (логическое умножение) |
| or | дизъюнкция (логическое сложение) |
| not | отрицание (инверсия) |

# Логический тип данных

```
program  n_7;
  var a, b, c: integer; ans: boolean;
begin
  writeln ('Определение истинности высказывания
                    о равнобедренном треугольнике');
  write ('Введите значения a, b, c>>');
  readln (a, b, c);
  ans:=(a=b) or (a=c) or (b=c);
  writeln ('Треугольник с длинами сторон ', a, ',', b,
                  ',', c, ' является равнобедренным - ', ans)
end.
```

# НАЧАЛА ПРОГРАММИРОВАНИЯ


## Программирование разветвляющихся алгоритмов

- условный оператор
- неполный условный оператор
- составной оператор
- вложенные ветвления

9 класс

ИЗДАТЕЛЬСТВО БИНОМ

# Общий вид условного оператора

Полная форма условного оператора:

**if** <условие> **then** <оператор_1> **else** <оператор_2>

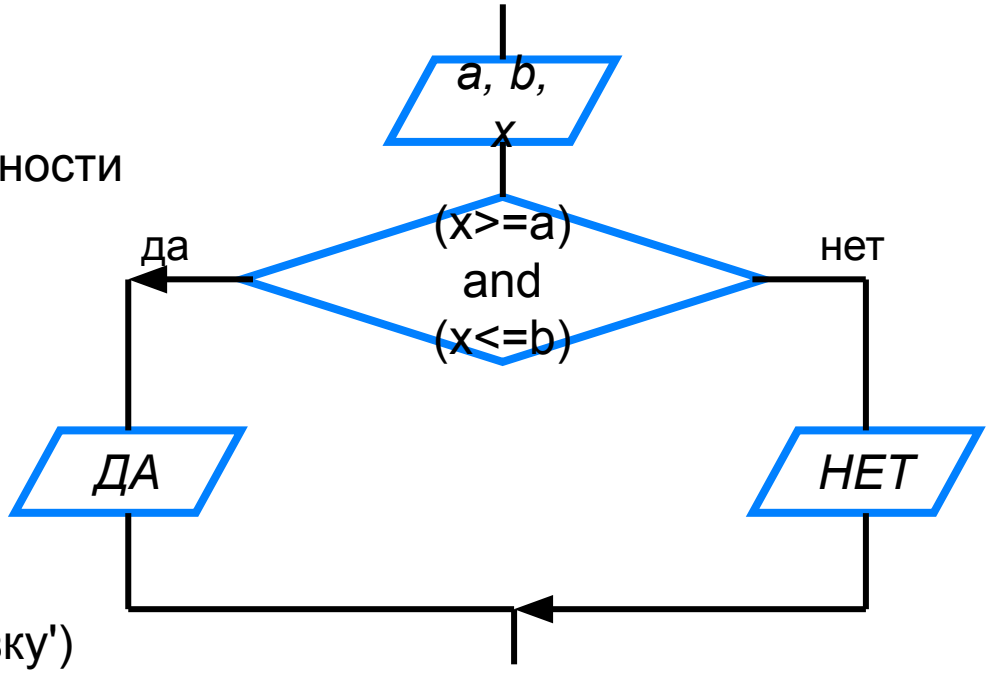
Неполная форма условного оператора:

**if** <условие> **then** <оператор>

! Перед **else** знак «;» не ставится.

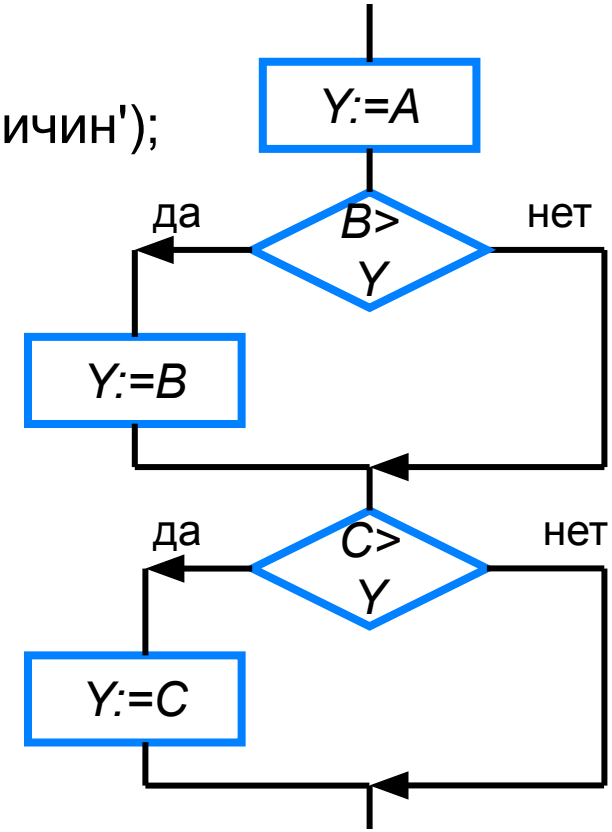
# Условный оператор

```
program n_9;
  var x, a, b: real;
begin
  writeln ('Определение принадлежности
                    точки отрезку');
  write ('Введите a, b>>');
  readln (a, b);
  write ('Введите x>>');
  readln (x);
  if (x>=a) and (x<=b) then
  writeln ('Точка принадлежит отрезку')
  else writeln ('Точка не принадлежит отрезку')
end.
```



# Неполная форма условного оператор

```
program n_10;
  var y, a, b, c: integer;
begin
  writeln ('Нахождение наибольшей из трёх величин');
  write ('Введите a, b, c>>');
  readln (a, b, c);
  y:=a;
  if (b>y) then y:=b;
  if (c>y) then y:=c;
  writeln ('y=', y)
end.
```



# Составной оператор

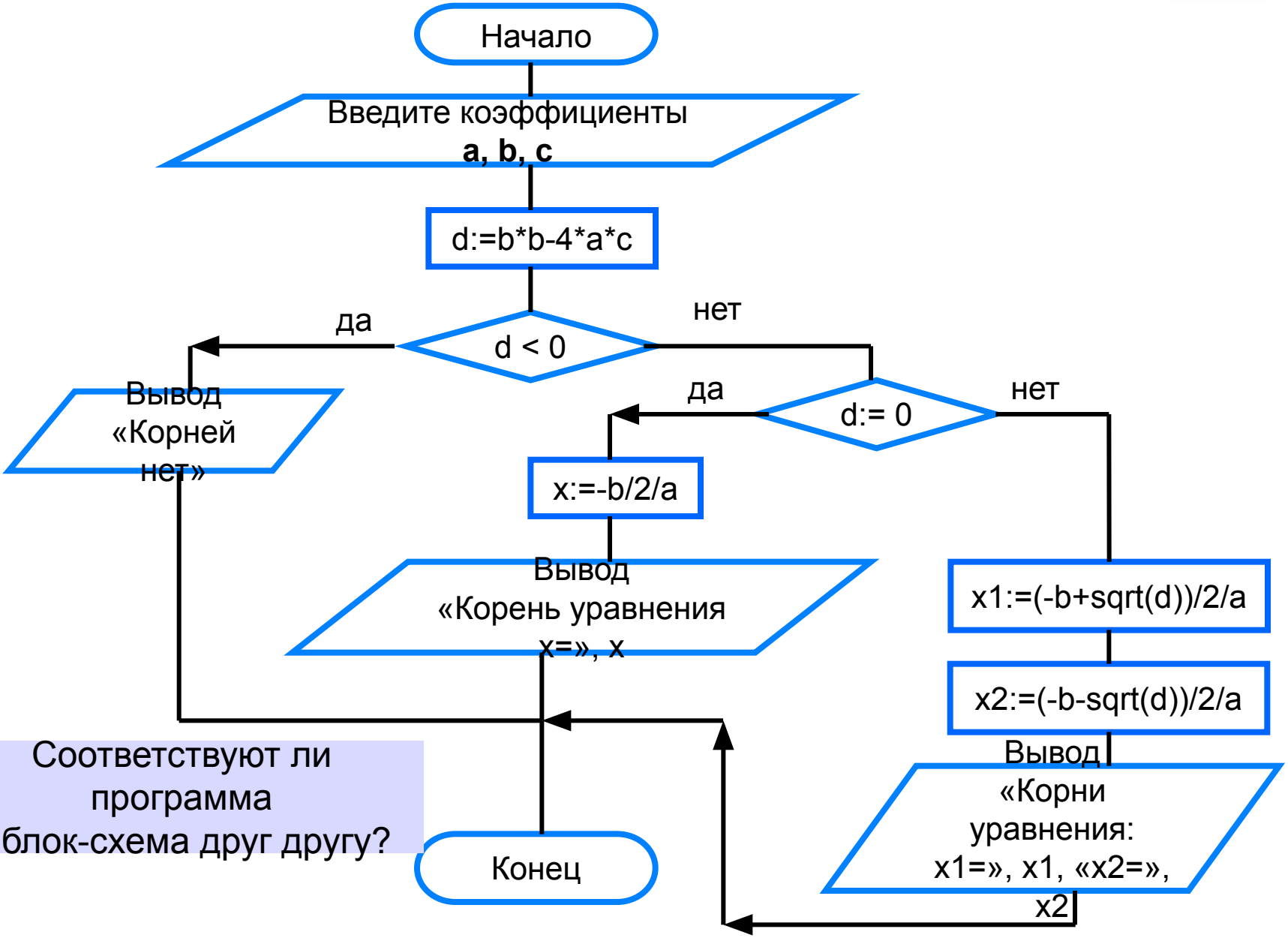
В условном операторе и после **then**, и после **else** можно использовать **только один оператор**.

Если в условном операторе после **then** или после **else** нужно выполнить **несколько операторов**, то используют **составной оператор** – конструкцию вида:

**begin** <последовательность операторов> **end**

```
program n_11;
  var a, b, c: real;
  var d: real;
  var x, x1, x2: real;
begin
  writeln ('Решение квадратного уравнения');
  write ('Введите коэффициенты a, b, c  >>');
  readln (a, b, c);
  d:=b*b-4*a*c;
  if d<0 then writeln ('Корней нет');
  if d=0 then
        begin
          x:=-b/2/a;
          writeln ('Корень уравнения x=', x:9:3)
        end;
    if d>0 then
        begin
          x1:=(-b+sqrt(d))/2/a;
          x2:=(-b-sqrt(d))/2/a;
          writeln ('Корни уравнения:');
           writeln ('x1=', x1:9:3);
          writeln ('x2=', x2:9:3)
        end;
  end.
```

# Блок-схема решения КВУР




? Соответствуют ли программа и блок-схема друг другу?

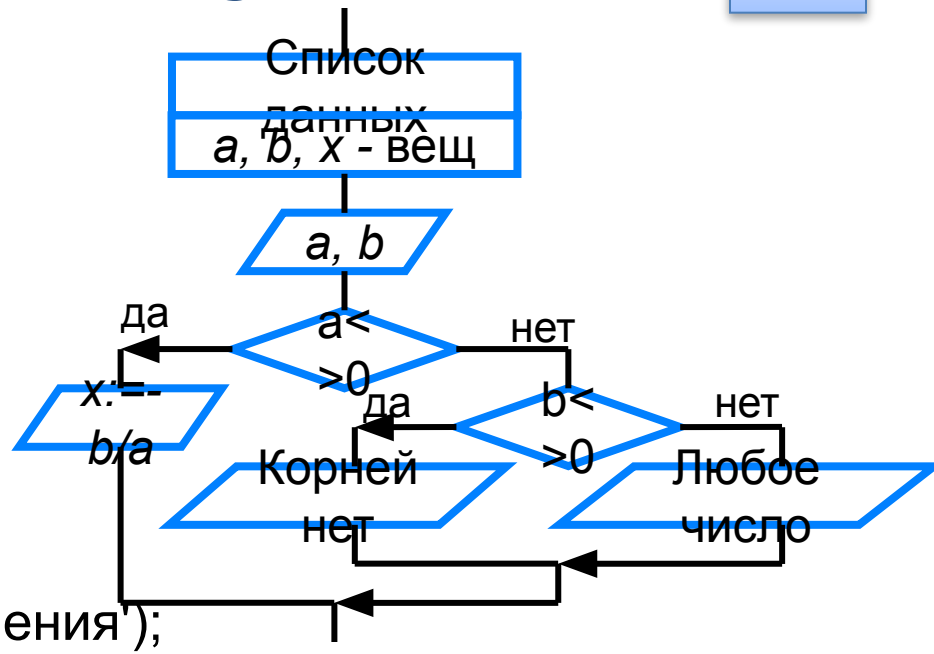
# Вложенные ветвления

Возможна следующая конструкция:

```
if <условие1> then
                    if <условие2> then
                      <оператор1>
                                    else <оператор2>
          else <оператор3>
```

! **else** всегда относится к ближайшему оператору **if**

# Решение линейного уравнения




```
program n_12;
  var a, b, x: real;
begin
  writeln ('Решение линейного уравнения');
  write ('Введите коэффициенты a , b>>');
  readln (a, b);
 if a<>0 then
              begin
                x:=-b/a;
                writeln ('Корень уравнения x=', x:9:3)
              end
            else if b<>0 then writeln ('Корней нет')
                         else writeln ('x – любое число');
end.
```

# НАЧАЛА ПРОГРАММИРОВАНИЯ

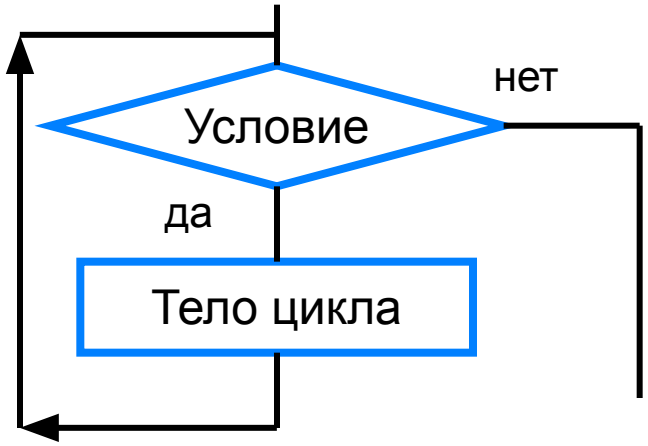

## Программирование циклических алгоритмов

- **while** ( цикл - ПОКА)
- **repeat** (цикл - ДО)
- **for** (цикл с параметрами)

9 класс

ИЗДАТЕЛЬСТВО БИНОМ

# Программирование циклов с заданным условием продолжения работы



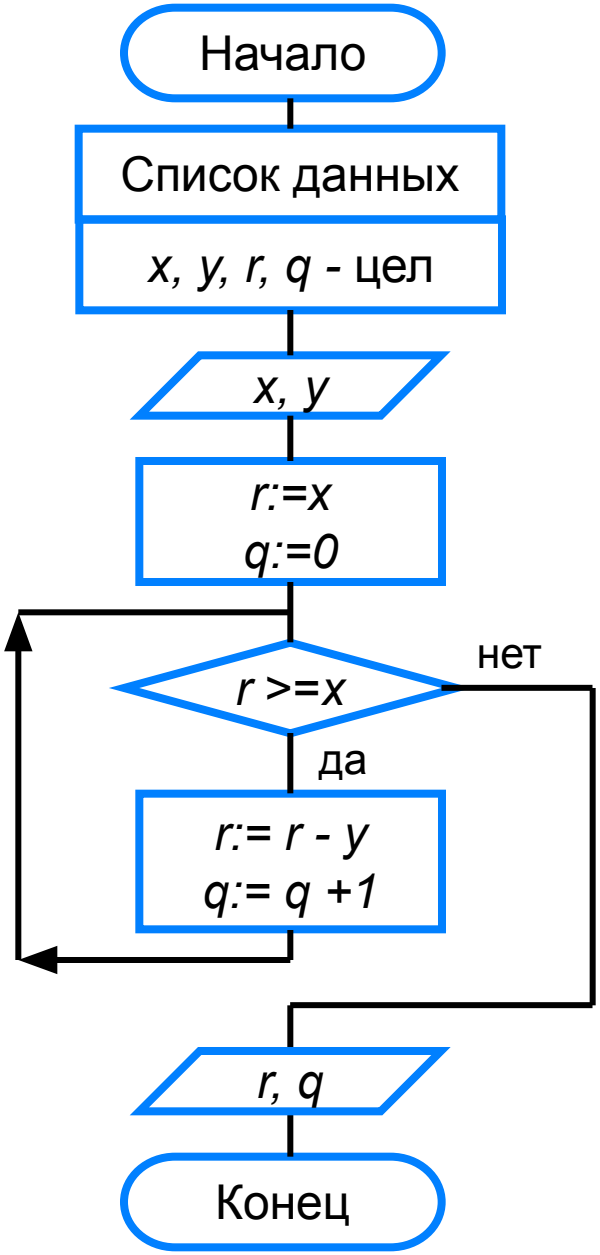

Общий вид оператора:

**while** <**условие**> **do** <**оператор**>

Здесь:

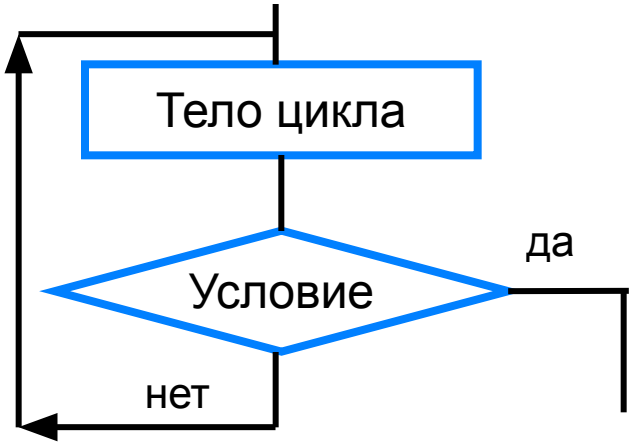
<**условие**> - логическое выражение; пока оно истинно, выполняется тело цикла;

<**оператор**> - простой или составной оператор, с помощью которого записано тело цикла.




```
program n_14;
  var x, y, q, r: integer;
begin
  writeln ('Частное и остаток');
  write ('Введите делимое x>>');
  readln (x);
  write ('Введите делитель y>>');
  read (y);
  r:=x;
  q:=0;
  while r>=x do
  begin
   r:=r-y;
   q:=q+1
  end;
  writeln ('Частное q=', q);
  writeln ('Остаток r=', r)
end.
```

# Программирование циклов с заданным условием окончания работы




Общий вид оператора:

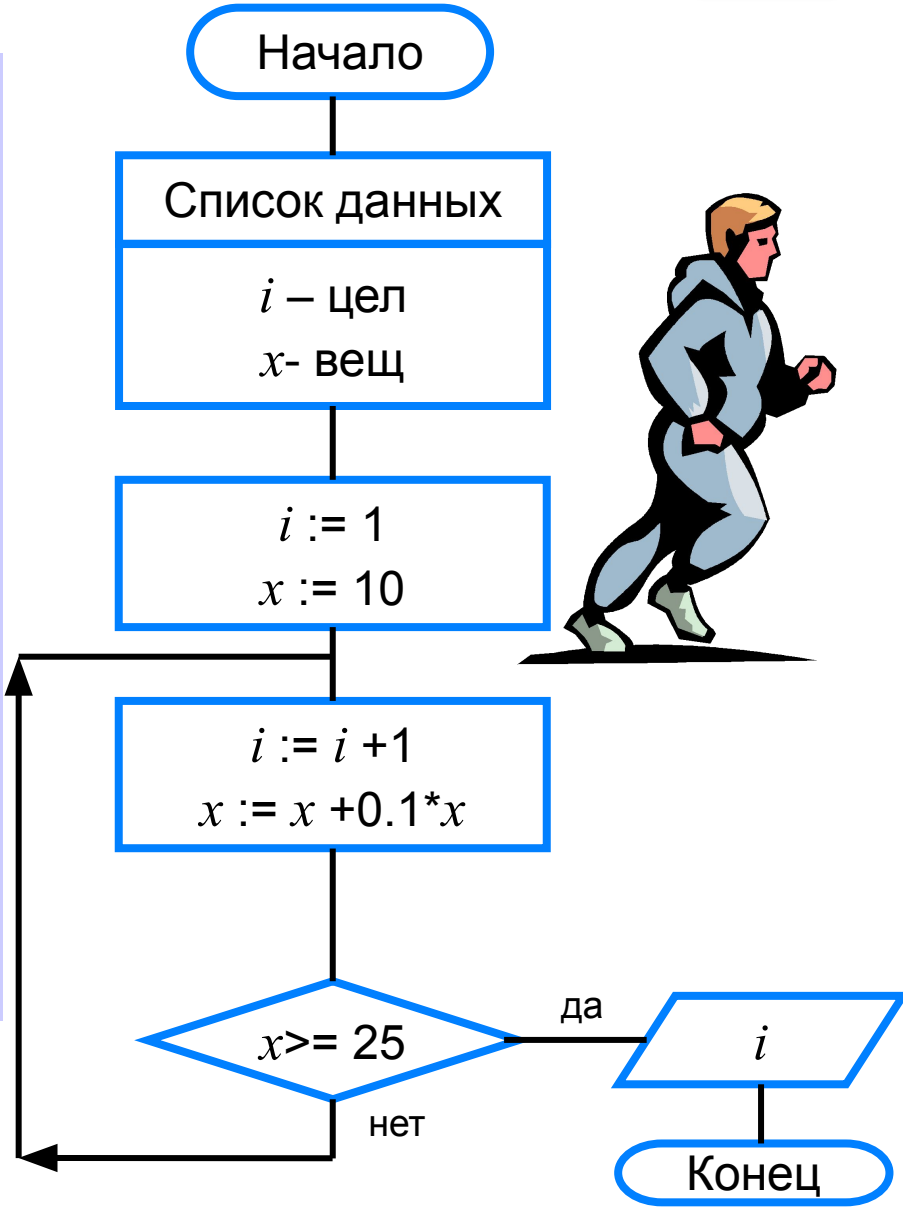
**repeat** <**оператор1; оператор2; …; >** **until** <**условие**>

Здесь:

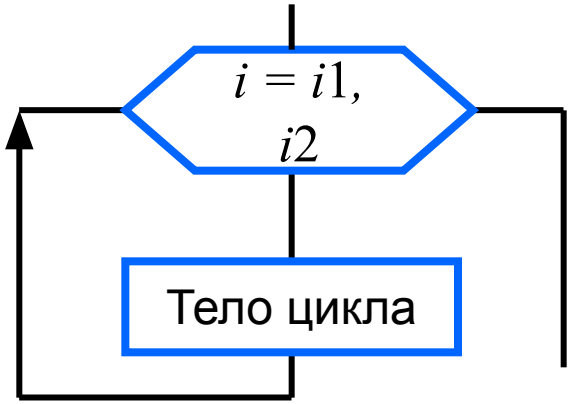
<**оператор1**>; <**оператор2**>; … - операторы, образующие тело цикла;

<**условие**> - логическое выражение; если оно ложно, то выполняется тело цикла.

```
program n_15;
  var i: integer; x: real;
begin
  writeln ('График тренировок');
  i:=1;
  x:=10;
  repeat
  i:=i+1;
  x:=x+0.1*x;
  until x>=25;
  writeln ('Начиная с ', i, '-го  дня спортсмен будет пробегать 25 км')
end.
```



# Программирование циклов с заданным числом повторений




Общий вид оператора:

**for** <**параметр**>:=<**начальное_значение**>

**to** <**конечное_значение**> **do** <**оператор**>

Здесь:

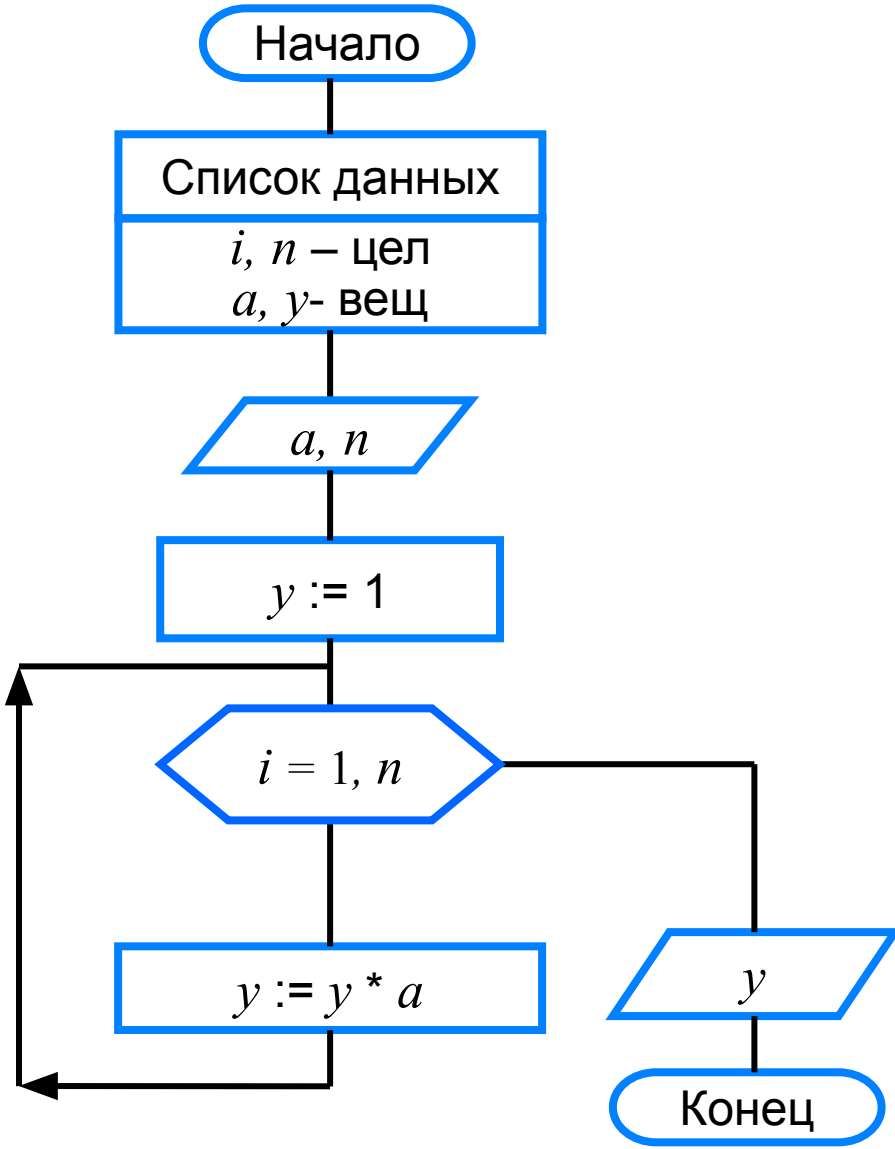
<**параметр**> - переменная целого типа;

<**начальное_значение**> и <**конечное_значение**> - выражения того же типа, что и параметр;

<**оператор**> - простой или составной оператор - тело цикла.

После каждого выполнения тела цикла происходит увеличение на единицу параметра цикла; условие выхода из цикла - превышение параметром конечного значения.

```
program n_16;
  var i,n:integer;a,y:real;
begin
  writeln ('Возведение в степень');
  write ('Введите основание a>>');
  readln (a);
  write ('Введите показатель n>>');
  readln (n);
  y:=1;
  for i:=1 to n do y:=y*a;
  writeln ('y=', y)
end.
```



# Различные варианты программирования циклического алгоритма

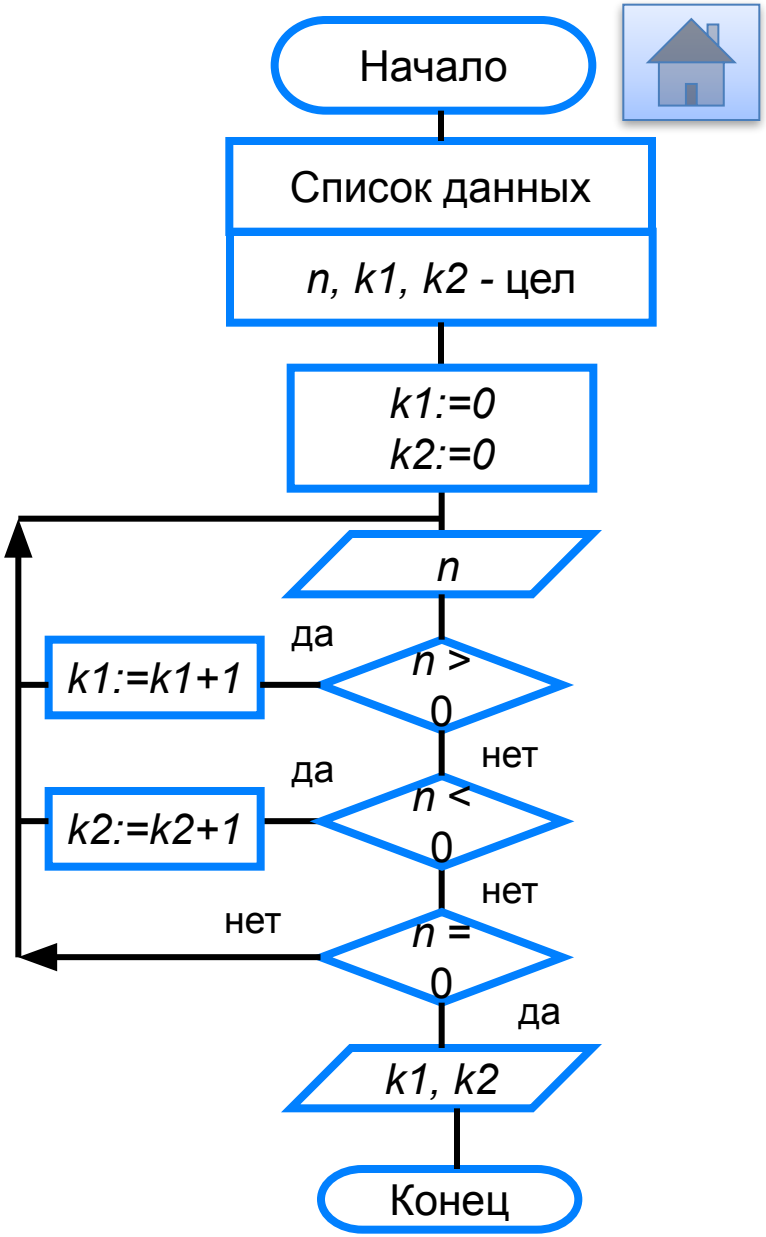
Для решения одной и той же задачи могут быть созданы разные программы.

Организуем ввод целых чисел и подсчёт количества введённых положительных и отрицательных чисел. Ввод должен осуществляться до тех пор, пока не будет введён ноль.

В задаче в явном виде задано условие окончания работы.

Воспользуемся оператором **repeat**.

```
program n_17;
  var n, k1, k2: integer;
begin
  k1:=0;
  k2:=0;
  repeat
    write ('Введите целое число>>');
    readln (n);
    if n>0 then k1:=k1+1;
    if n<0 then k2:=k2+1;
  until n=0;
  writeln ('Введено:');
  writeln ('положительных чисел – ', k1);
  writeln ('отрицательных чисел – ', k2)
end.
```



Ввод осуществляется до тех пор, пока не будет введён ноль.

Работа продолжается, пока ***n*** ≠0.

Воспользуемся оператором **while**:

```
program n_18;
  var n, k1, k2: integer;
begin
  k1:=0;
  k2:=0;
  n:=1;
  while n<>0 do
  begin
    writeln ('Введите целое число>>');
    read (n);
    if n>0 then k1:=k1+1;
    if n<0 then k2:=k2+1;
  end;
  writeln ('Введено:');
  writeln ('положительных – ', k1);
  writeln ('отрицательных – ', k2)
end.
```