Разветвляющиеся вычислительные процессы

Наумов Д.А., доц. каф. КТ

Программирование и алгоритмические языки, 2019

Содержание лекции

Разветвтляющиеся алгоритмы

Разветвляющийся вычислительный процесс

в зависимости от выполнения определенных условий реализуется по одному из нескольких заранее предусмотренных направлений.

- Каждое отдельное направление называется - **ветвью вычислений**.
- Выбор той или иной ветви осуществляется уже **при выполнении программы** в результате проверки **некоторых условий** и определяется свойствами исходных данных и промежуточных результатов.

$$y = \begin{cases} \frac{1}{2}\sqrt[4]{x}, & \text{если } x > 1 \\ \frac{1}{3}\sqrt[3]{x}, & \text{если } 0 < x \leq 1 \\ \frac{1}{4}\sqrt[4]{x}, & \text{если } x \leq 0 \end{cases}$$

Тип Boolean

Для реализации разветвляющих алгоритмов в языке Pascal предусмотрен специальный тип данных - *логический*. Переменная логического типа описывается следующим образом:

```
1 var
2   ИмяПеременной: Boolean;
```

Переменная логического типа может принимать одно из двух значений:

- логическая ложь (константа логического типа False);
- логическая истина (константа логического типа True);

Значение типа Boolean можно вывести операторами write/writeln, но нельзя считать операторами read/readln.

Операции отношения

Для задания условия в логическом выражении используются операции отношения.

Операции отношения

Операция отношения – это конструкция вида $A\Theta B$, где

- A и B – любые выражения языка ActionScript,
- Θ – знак операции отношения.

Допустимы следующие операции отношения:

- $<$ (меньше),
- $<=$ (меньше или равно),
- $>$ (больше),
- $>=$ (больше или равно),
- $=$ (равно),
- $<>$ (не равно)

Пример

- результат вычисления выражения 4 > 5 равен false;
- результат вычисления выражения 2 >= 2 равен true;
- результат вычисления выражения 9 !=0 равен true;

```
1 var x, y: real;
2 begin
3  x := 0.1;
4  x := x + 0.1; x := x + 0.1;
4  x := x + 0.1; x := x + 0.1;
5  y := 1/2;
6  writeln(x=y); //результат false!
7  readln;
8 end.
```

Пример. Сравнение двух вещественных чисел

- вводится величиная погрешности, которая будет использоваться при сравнении чисел;
- два числа будут считаться равными (приблизительно), если модуль их разности не превышает погрешность.

$$|X - Y| < \varepsilon$$

```
1 const eps = 1e-6;
1 var x, y: real;
2 begin
3   x := 0.1;
4   x := x + 0.1; x := x + 0.1;
4   x := x + 0.1; x := x + 0.1;
5   y := 1/2;
6   writeln(abs(x-y)<eps); //результат true
7   readln;
8 end.
```

Логические операции

Для объединения нескольких логических выражений использу-ют логические операции:

- not A - отрицание;
- A and B - конъюнкция (логическое умножение);
- A or B - дизъюнкция (логическое сложение);
- A xor B - сложение по модулю 2 (исключающее или);

A и B - выражения логического типа, тип результата - логический.

Конъюнкция (AND)

a	b	$a \wedge b$
0	0	0
1	0	0
0	1	0
1	1	1

Дизъюнкция (OR)

a	b	$a \vee b$
0	0	0
1	0	1
0	1	1
1	1	1

Сложение по модулю 2 (XOR)

a	b	$a \oplus b$
0	0	0
1	0	1
0	1	1
1	1	0

Отрицание (NOT)

a	$\neg a$
0	1
1	0

Пример. Сравнение двух вещественных чисел

- вводится величиная погрешности, которая будет использоваться при сравнении чисел;
- два числа будут считаться равными (приблизительно), если модуль их разности не превышает погрешность.

$$|X - Y| < \varepsilon$$

```
1 const eps = 1e-6;
1 var x, y: real;
2 begin
3   x := 0.1;
4   x := x + 0.1; x := x + 0.1;
4   x := x + 0.1; x := x + 0.1;
5   y := 1/2;
6   writeln(abs(x-y)<eps); //результат true
7   readln;
8 end.
```

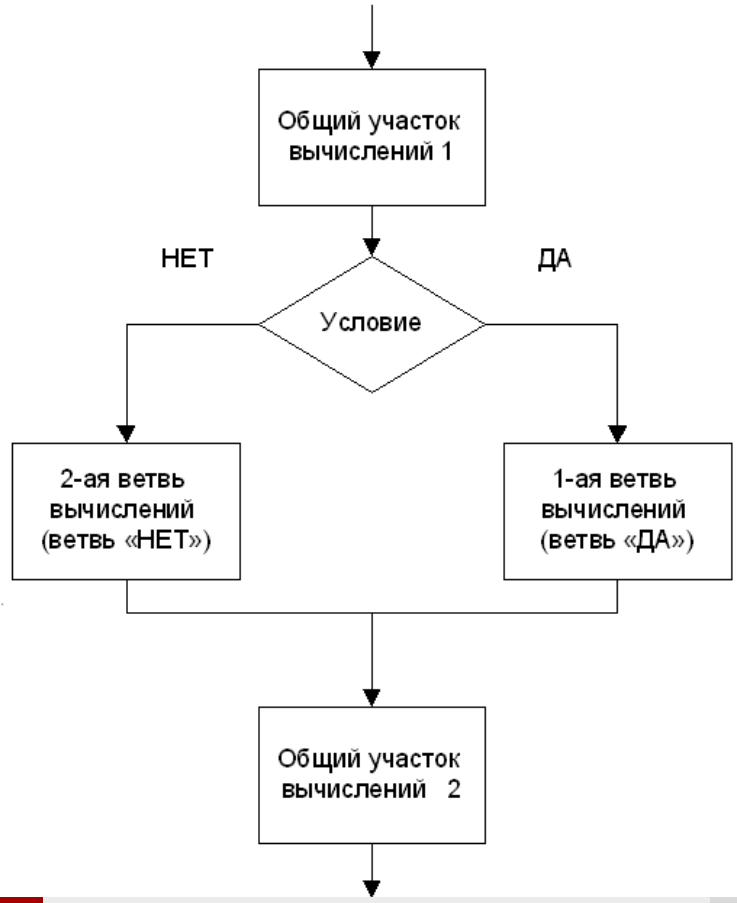
Приоритет операций

- действия в скобках;
- *not*;
- $*, /, div, mod, and, shr, shl$;
- $+, -, or, xor$;
- $=, <>, >, <, >=, <=$;

```
1 var x, y: integer;
2 begin
3   x = 8; y = -1;
4   writeln(0 < x < 10);              // ошибка
5   writeln(0 < x and x < 10);        // ошибка
6   writeln((0 < x) and (x < 10));    // ошибка
7   writeln((0 < x) and (x < 10));    // нет ошибки
8   writeln(x and y);                 // ?
9 end.
```

Разветвляющийся вычислительный процесс, содержащий две ветви, схематично может быть изображен с помощью структуры выбора (структуры разветвления), которая содержит три элемента:

- логическое условие,
- ветвь «ДА»,
- ветвь «НЕТ».



Условный оператор

```
1 if выражение then
2    оператор1
3 else
4    оператор2;
```

- выражение - выражение *логического типа*;
- оператор1, оператор2 - (одиночные) операторы языка Паскаль.

Порядок выполнения условного оператора:

1. вычисляется значение *логического выражения*;
2. если значение логического выражения равно **true**, то выполняется Оператор1 (а Оператор2 пропускается);
3. если значение логического выражения равно **false**, то выполняется Оператор2 (а Оператор1 пропускается);
4. выполняется оператор, стоящий в программе непосредственно после оператора *if*.

Пример

Вычисление функции по одной из двух предложенных формул в зави-симости от значения аргумента

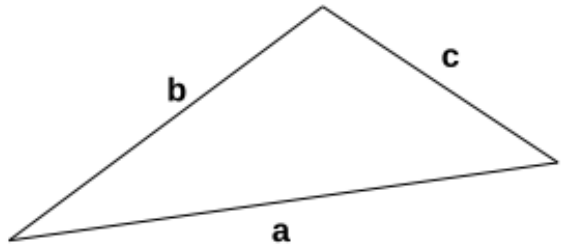
$$y = \begin{cases} x^2, & \text{если } x < 0 \\ \sqrt{x}, & \text{если } x \geq 0 \end{cases}$$

Оператор, реализующий эти вычисления для некоторого значения аргумента x, выглядит следующим образом:

```
1 if x < 0 then
2    y := x * x
3 else
4    y := sqrt(x);
```

Пример

Задача для определения, можно ли построить треугольник из отрезков заданной длины: x, y, z ($x > 0$, $y > 0$, $z > 0$).



Оператор, реализующий эти вычисления для некоторого значения аргумента x, выглядит следующим образом:

```
1  if (x +y > z) and (x +z > y) and (y +z > x)
2    writeln('треугольник построить можно')
3  else
4    writeln('треугольник построить нельзя');
```

Сокращенная форма условного оператора

```
1 if выражение then
2     оператор1;
```

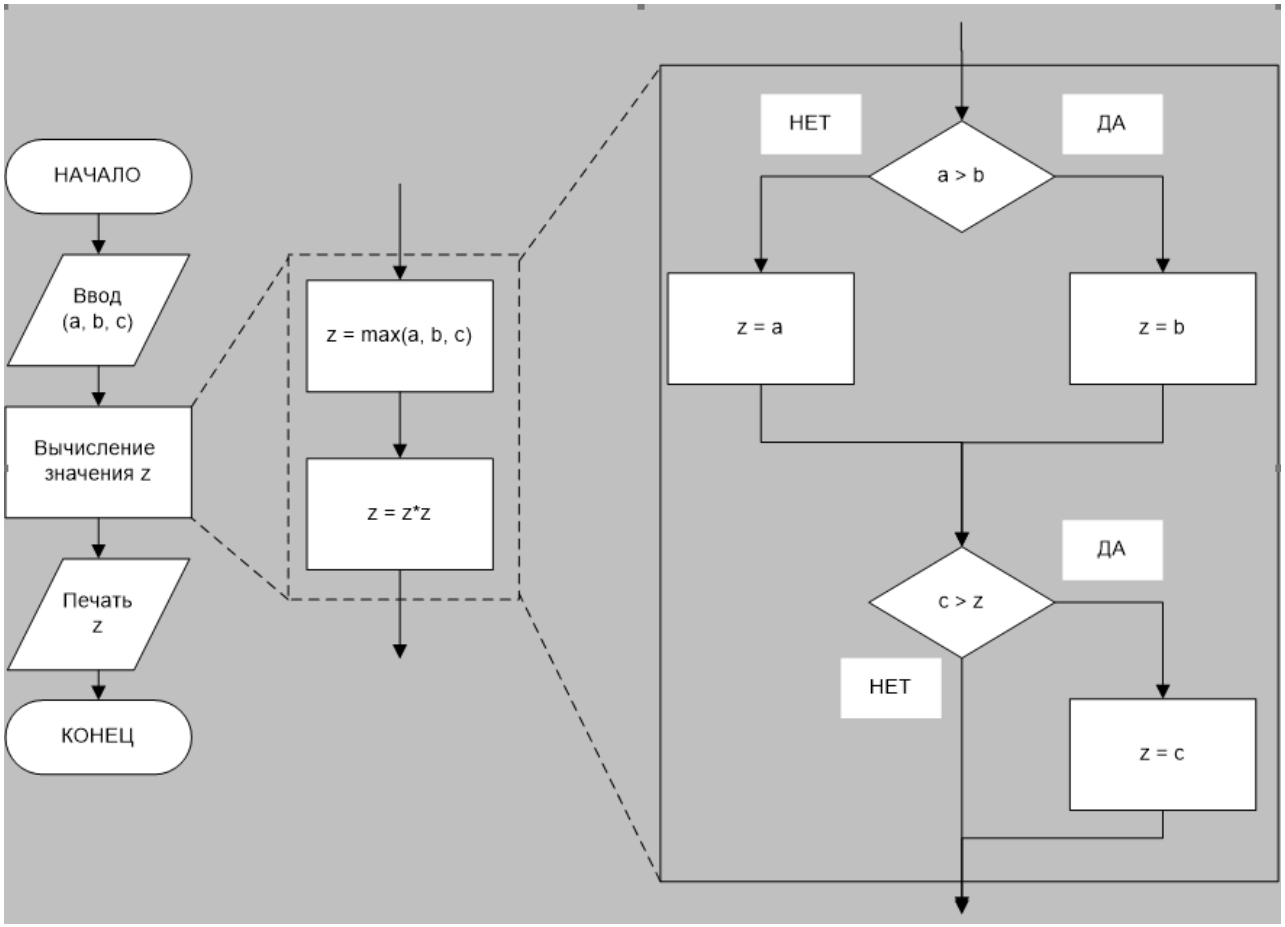
- выражение - выражение *логического типа*;
- оператор1 - (одиночный) оператор языка Паскаль.

Порядок выполнения условного оператора:

1. вычисляется значение *логического выражения*;
2. если значение логического выражения равно **true**, то выполняется Оператор1;
3. выполняется оператор, стоящий в программе непосредственно после оператора *if*.

Пример

Даны три неравных числа a, b, c. Вычислить и напечатать значение z, равное квадрату большего из них.



Вложенные операторы if

Условные операторы могут иметь вложенную конструкцию, когда в качестве Оператора1 или Оператора2 может также использоваться условный оператор.

Правило для вложенных операторов

ключевое слово **else** всегда относится к ближайшему предыдущему оператору **if**.

```
1 z := 0;
2 if x < 0 then
3    if y < 0 then
4       z := 1
5 else
6    z := 2; // в каких случаях z будет равно 2?
```

Пример

Вычислить значение функции по одной из предложенных формул.

$$y = \begin{cases} \frac{1}{2}\sqrt{x}, & если\ x > 1 \\ \frac{1}{3}\sqrt[3]{x}, & если\ 0 < x \le 1 \\ \frac{1}{4}\sqrt[4]{|x|}, & если\ x \le 0 \end{cases}$$

```
1 if x > 1 then
2    y := 1/2 * sqrt(x)
3 else
4   if x > 0 then
5     y := 1/3 * exp(1/3*ln(x))
6   else
7     y := 1/4 * exp(1/4*ln(abs(x)));
```

Составной оператор

В состав условного оператора может входить только один оператор. Если в какую-либо ветвь разветвления требуется вставить несколько операторов, то они объединяются в один, составной оператор:

```
1 begin
2   оператор1;
3   оператор2;
4   ...
5   операторN;
6 end
```

Элементами составного оператора могут быть любые операторы языка, в том числе и другие составные операторы.

Пример

Вычислить корни квадратного уравнения общего вида.

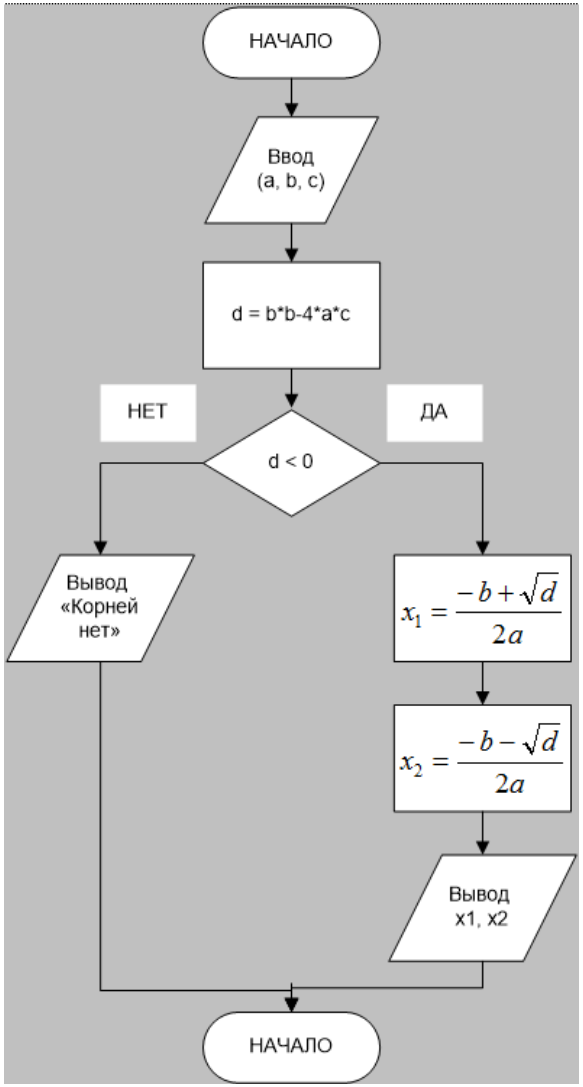
$$ax^2 + bx + c = 0, a \neq 0$$

Введем обозначения:

$$x_1 = \frac{-b - \sqrt{d}}{2a}, x_2 = \frac{-b + \sqrt{d}}{2a}$$

- A, B, C - коэффициенты уравнения;
- D - дискриминант;
- X1, X2 -корни уравнения.

Блок-схема


НАЧАЛО
Ввод
(a, b, c)
d = b*b-4*a*c
НЕТ
ДА
d < 0
Вывод
«Корней
нет»
$x_1 = \frac{-b+\sqrt{d}}{2a}$
$x_2 = \frac{-b-\sqrt{d}}{2a}$
Вывод
x1, x2
НАЧАЛО

Фрагмент программы

```
//расчет определителя
d := b*b - 4*a*c;

//вывод промежуточных результатов
if DEBUG then
   writeln('  d = ', d);

writeln;
if d < 0 then
  writeln('Действительных корней нет');
else
  begin
    x1 := (-b - sqrt(d))/(2*a);
    x2 := (-b + sqrt(d))/(2*a);

    writeln('Значения корней уравнения');
    writeln('  x1 = ', x1:10:4);
    writeln('  x2 = ', x2:10:4);
  end;
```