3. ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

Цель работы: изучение программирования разветвляющихся алгоритмов, логического типа данных, логических операций и операций отношения, условного и составного операторов в языке Pascal.

Методические указания

Вычислительный процесс называется разветвляющимся, если в зависимости от выполнения определенных условий он 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.

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

```
var ИмяПеременной: Boolean;
```

Переменная логического типа может принимать одно из двух значений: логическая ложь и логическая истина. Для этих значений заданы логические константы с идентификаторами **false** и **true**.

Для задания условия в логическом выражении используются операции отношения. Операция отношения – это конструкция вида AΘB, где A и B – любые выражения языка, Θ – знак операции отношения. Допустимы следующие операции отношения: < (меньше), <= (меньше или равно), > (больше), >= (больше или равно), = (равно), <> (не равно).

Результат вычисления операции отношения – значение логического типа данных.

Пример:

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

Для объединения нескольких логических выражений используют логические операции:

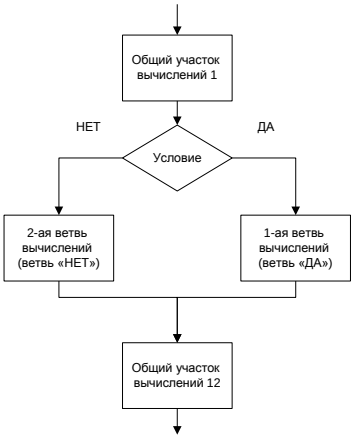
- not (логическое отрицание),
- and (логическое умножение),
- or (логическое сложение).

Операция логического отрицания применяется к одному операнду. Если значение операнда было истинным, то его отрицание - ложь, и наоборот.

Операция конъюнкции **and** применяется к двум операндам. Операция будет давать значение логическая истина, если оба операнда имеют значение истина, в противном случае результат данной операции - логическая ложь.

Операция дизъюнкции **or** применяется к двум операндам. Операция будет давать значение логическая истина, если хотя бы один из операндов имеет значение истина, в противном случае результат данной операции - логическая ложь.

Операция сложения по модулю два **xor** применяется к двум операндам. Операция будет давать значение логическая истина, если аргументы различны, в противном случае результат данной операции - логическая ложь.



Разветвляющийся вычислительный процесс, содержащий две ветви, схематично может быть изображен с помощью структуры выбора (структуры разветвления), которая содержит три элемента: логическое условие, ветвь "ДА" и ветвь "НЕТ".

После вычислений, общих для обоих ветвей, проверяется некоторое условие. Если условие выполняется, то осуществляется переход к ветви "ДА", в противном случае - к ветви "НЕТ". После выполнения вычислений в любой из ветвей осуществляется переход к общему участку.

Структура выбора реализуется с помощью условного оператора (оператора условного перехода), который позволяется выполнить один из двух входящих в него операторов в зависимости от значения некоторого логического выражения.

Оператор имеет следующий вид:

```
if Логическое выражение then
    Оператор1
else
    Оператор2;
```

где **if** и **else** - служебные слова, Оператор1 и Оператор2 - любые операторы языка.

Порядок выполнения условного оператора следующий:

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

Простейший пример использования условного оператора - это вычисление функции по одной из двух предложенных формул в зависимости от значения аргумента:

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

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

```
if x < 0 then
  y := x*x;
else
  y := sqrt(x);
```

Примером, когда логическое выражение в операторе **if** имеет более сложную структуру, может служить задача для определения, можно ли построить треугольник из отрезков заданной длины: x, y, z ($x > 0, y > 0, z > 0$). Условный оператор имеет вид:

```
if (x +y > z) and (x +z > y) and (y +z > x) then
  writeln('треугольник построить можно');
else
  writeln('треугольник построить нельзя');
```

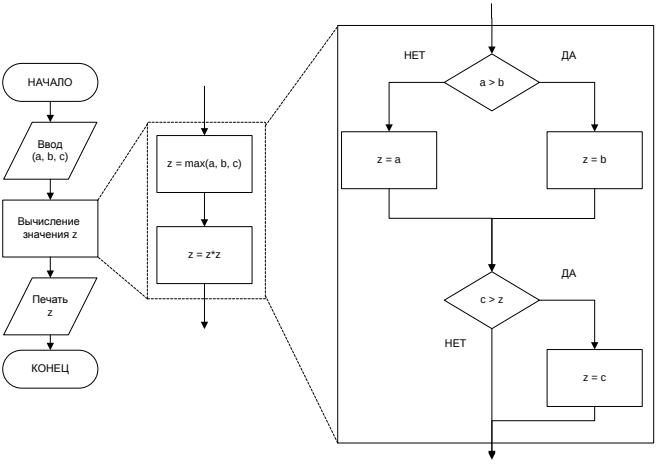
Условный оператор может не иметь конструкции **else**, такая форма оператора называется сокращенной:

```
if Логическое выражение then
    Оператор1;
```

Порядок выполнения условного оператора в сокращенной форме следующий:

1) если значение логического выражения равно **true**, то выполняется **Оператор1**;
2) если значение логического выражения равно **false**, то **Оператор1** пропускается;
3) далее 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.



Построение схематического описания алгоритма решения задачи с постепенным уточнением и детализацией блоков представлено на рисунке.

Условные операторы могут иметь вложенную конструкцию, когда в качестве **Оператора1** или **Оператора2** может также использоваться составной оператор. При этом справедливо следующее правило: **else** всегда относится к ближайшему предыдущему оператору **if**.

Пример: требуется вычислить значение функции по одной из предложенных формул:

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

Для программной реализации этих вычислений можно использовать вложенную конструкцию условного оператора:

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

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*:

```
begin
  Оператор1;
  Оператор2;
  ...
  ОператорN;
end
```

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

Пример: требуется вычислить корни квадратного уравнения общего вида:

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

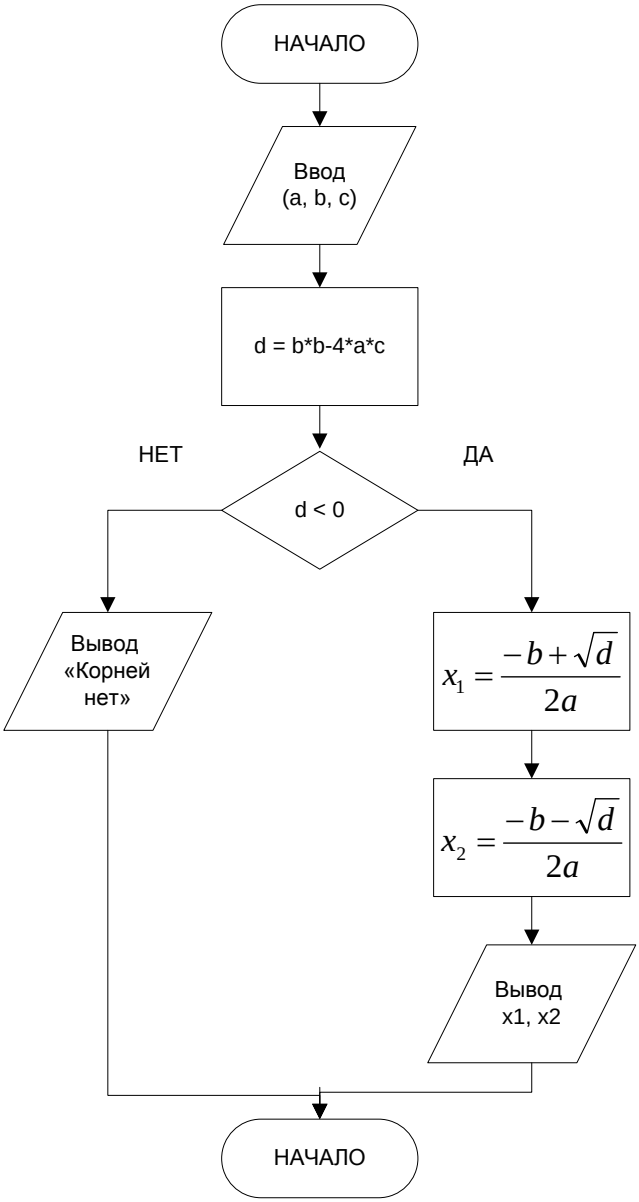
Введем следующие обозначения:

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

A, B, C - коэффициенты уравнения;

D - дискриминант;

X1, X2 -корни уравнения.


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

Программа для нахождения корней квадратного уравнения будет иметь вид:

```
program lec02_ex01_equation;

const
  DEBUG = true; //выводить ли на экран
                //промежуточные результаты

var
  //исходные данные
  a, b, c, d: double;
  //результат вычислений
  x1, x2: double;

begin
  //ввод исходных данных
  writeln('Нахождение корней квадратного
уравнения');
  writeln('  ax^2+bx+c=0, a<>0');
  writeln('Введите значения коэффициентов a, b,
c');
  write('  a = ');
  readln(a);
  write('  b = ');
  readln(b);
  write('  c = ');
  readln(c);

  //эхо-печать исходных данных
  writeln('Введены значения');
  writeln('  a = ', a:10:4);
  writeln('  b = ', b:10:4);
  writeln('  c = ', c:10:4);

  //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;

  writeln('Для завершения работы нажмите Enter');
  readln;
end.
```

Контрольные вопросы

1. Что такое разветвляющийся вычислительный процесс?
2. Логический тип данных в языке ActionScript.
3. Операции отношения.
4. Логические операции.
5. Условный оператор.
6. Полная форма и сокращенная форма условного оператора.
7. Составной оператор. Назначение составного оператора.

Упражнения для самостоятельной работы

Составьте блок-схему алгоритма и напишите программу для вычисления значения функции для произвольного значения аргумента.

1) $$y = \begin{cases} \dfrac{1}{\sin(x)+2}, \text{ если } x < 0 \\ x^{-2}, \text{ если } 1 > x \geq 0 \\ x^{15}, \text{ если } 1 < x \end{cases};$$

2) $$y = \begin{cases} 0, \text{ если } x < -\pi \\ |\sin(x)|, \text{ если } -\pi \leq x \leq \pi \\ 0, \text{ если } \pi \leq x \leq \dfrac{3}{2}\pi \\ x - \dfrac{3}{2}\pi, \text{ если } \dfrac{3}{2}\pi \leq x \end{cases}$$

Задание к лабораторной работе

Задание 1. Разработайте **алгоритм** для решения следующих задачи:

№ варианта	Задание
1	Выведите значения переменных a, b и c в порядке возрастания их значений.
2	Даны отрезки a, b и c. Составьте программу, определяющую, можно ли из них построить треугольник.
3	Выведите значения переменных a, b и c в порядке убывания их значений.
4	Даны отрезки a, b и c. Составьте программу, определяющую, можно ли из них построить равнобедренный треугольник.
5	Составьте программу нахождения произведения двух наибольших из трех чисел a, b и c.
6	Даны отрезки a, b и c. Составьте программу, определяющую, можно ли из них построить равносторонний треугольник.
7	Составьте программу нахождения произведения двух наименьших из трех чисел a, b и c.
8	Даны отрезки a, b и c. Составьте программу, определяющую, можно ли из них построить прямоугольный треугольник.
9	Составьте программу определения номера наибольшего из трех чисел x1, x2 и x3.
10	Даны отрезки a, b, c и d. Составьте программу, определяющую, можно ли из них построить ромб.
11	Составьте программу определения номера наименьшего из трех чисел x1, x2 и x3.

Задание 2. Составьте блок-схему алгоритма и напишите программу для решения следующих задачи.

№ варианта	Задание
1	Определить, к каком квадранте координатной плоскости лежит точка координатами (x, y)
2	Определить принадлежность точки на плоскости с ко-

	ординатами (x, y) кругу радиуса R с центром в точке (x0, y0);
3	Определить принадлежность точки на плоскости с координатами (x, y) кольцу с внутренним радиусом r и внешним радиусом R с центром в точке (0, 0);
4	Определить принадлежность точки на плоскости с координатами (x, y) области, ограниченной прямыми x = -3, x = 0, y = 0, y = 5;
5	Определить принадлежность точки на плоскости с координатами (x, y) области, ограниченной осью абсцисс, прямой y = 1 и параболой y = x*x;
6	Определить принадлежность точки на плоскости с координатами (x, y) области, ограниченной осями координат и прямой y = x + a.
7	Определить принадлежность точки на плоскости с координатами (x, y) области, ограниченной второй координатной четвертью и кругом радиуса R с центром в начале координат
8	Определить принадлежность точки на плоскости с координатами (x, y) области, ограниченной прямыми y = 2x, y = x/2, y = 1/x
9	Определить принадлежность точки на плоскости с координатами (x, y) области, ограниченной прямыми y = 0, x = 0, y = x + 1
10	Определить принадлежность точки на плоскости с координатами (x, y) области, ограниченной параболами $y = x^2$; $y = x^3$
11	Определить принадлежность точки на плоскости с координатами (x, y) области, ограниченной графиками функций y = x-1; y = ln(x)