Урок 3. Программирование ветвлений

Цель работы: приобрести навыки в решении задач с помощью условного оператора и оператора выбора, усвоить назначение и правила их применения.

Для программирования ветвящихся алгоритмов в языке С имеется несколько различных средств. К ним относятся рассмотренная выше операция условия ?: , условный оператор if и оператор выбора switch.

Условный оператор. Формат условного оператора следующий:

```
if(выражение) оператор1; else оператор2;
```

Это полная форма оператора, программирующая структуру полного ветвления. Обычно выражение — это некоторое условие, содержащее операции отношения и логические операции. Значение выражения приводится к целому и интерпретируется в соответствии с правилом: равно нулю — ложь, не равно нулю — истина.
Если выражение истинно, выполняется оператор1, если ложно — оператор2.
Необходимо обратить внимание на следующие особенности синтаксиса условного оператора:

- выражение записывается в круглых скобках;
- точка с запятой после оператора 1 ставится обязательно.

Возможно использование неполной формы условного оператора

```
if  (выражение) оператор;
```

Вот пример использования полной формы условного оператора для нахождения большего значения из двух переменных a и b:

```
if(a > b)  max = a;  else  max = b;
```

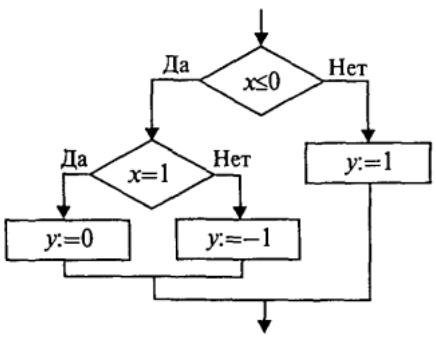
Та же самая задача может быть решена с использованием неполного ветвления следующим образом:

```
max = a;  if(b > a)  max = b;
```

Теперь рассмотрим примеры программирования вложенных ветвящихся структур. Требуется вычислить функцию sign(x) — знак x, которая определена следующим образом:

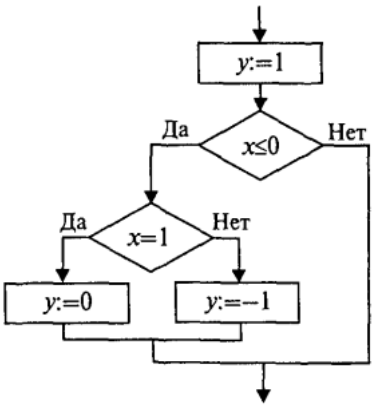
$$\text{sign}(x) = \begin{cases} -1, & \text{если } x < 0, \\ 0, & \text{если } x = 0, \\ 1, & \text{если } x > 1, \end{cases}$$

Пример 1. Алгоритм с полными вложенными ветвлениями:



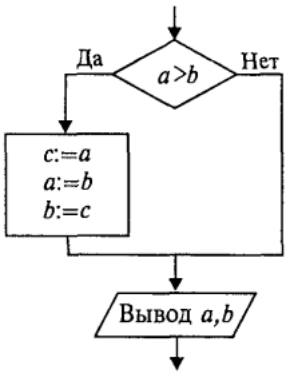

```
if(x <= 0)
      if(x == 0)  y = 0;
      else  y = -1;
else  y = 1;
```

Пример 2. Алгоритм с неполным ветвлением:




```
y = 1;
      if(x <= 0)
      if(x == 0)   y = 0;
else  y = -1;
```

Пример 3. Упорядочить по возрастанию значения в двух переменных a, b




```
if(a > b)
      {
      c = a;  a = b;  b = c;
      }
cout << " a = " << a << " b = " << b ;
```

В данном примере использован составной оператор — последовательность операторов, заключенная в фигурные скобки.

Обратите внимание на то, что перед закрывающей фигурной скобкой точку с запятой надо ставить обязательно, а после скобки точка с запятой не ставится.

В следующем примере вернемся к задаче вычисления площади треугольника по длинам трех сторон. Добавим в программу проверку условия правильности исходных данных: a, b, c должны быть положительными, а сумма длин каждой пары сторон треугольника должна быть больше длины третьей стороны.

Пример 4.

```
// Площадь треугольника

#include <iostream>
#include <cmath>
#include <clocale>

using namespace std;

void main( )
      {
      setlocale(LC_ALL, "RUS");
      float a, b, c, p, s;
      cout << "\na ="; cin >> a;
      cout << "\nb ="; cin >> b;
```

```
cout << "\nc ="; cin >> c;
if(a > 0 && b > 0 && c > 0 && a + b > c && a + c > b && b + c > a)
{
p = (a + b + c)/2;
s = sqrt(p *(p - a)*(p - b)*(p - c));
cout  <<"\nПлощадь  треугольника =" << s;
}
else  cout <<"\n Неверные исходные данные.";
}
```

Оператор выбора (переключатель). Формат оператора выбора:

```
switch  (целочисленное_выражение)
{
case константа1: список_операторов;
case константа2: список_операторов;
……
default:  список_операторов;
}
```

Последняя строка (default) может отсутствовать.
Выполнение оператора происходит в следующем порядке:

1. Вычисляется выражение.
2. Полученное значение последовательно сравнивается с константами, помещенными после служебного слова case; при первом совпадении значений выполняются операторы, стоящие после двоеточия.
3. Если ни с одной из констант совпадения не произошло, то выполнятся операторы после слова default.

Для того чтобы «обойти» выполнение операторов на последующих ветвях, нужно принять специальные меры, используя операторы выхода или перехода.
Рассмотрим фрагмент программы, который переводит числовую оценку знаний ученика в ее словесный эквивалент. Согласно системе оценок: 5 — «отлично», 4 — «хорошо», 3 — «удовлетворительно», 2 — «неудовлетворительно».

Пример 5.

```
#include  <iostream>
#include  <clocale>

using namespace std;

void main( )
{
setlocale(LC_ALL, "RUS");
int ball;
cout << "\nВведите  оценку: "; cin >> ball;
switch( ball )
      {
      case 2: cout << "\tЭто  неудовлетворительно!"; break;
      case 3: cout << "\tЭто  удовлетворительно!\n"; break;
      case 4: cout << "\tЭто  хорошо!\n";            break;
      case 5: cout << "\tЭто  отлично!\n";           break;
      default: cout << "\tНет такой оценки!\n";
      }
}
```

Здесь используется еще один новый для нас оператор break — оператор выхода. Его исполнение завершает работу оператора выбора, т.е. происходит «обход» других ветвей. Вот два варианта результатов выполнения этой программы:

```
Введите оценку: 3 Это удовлетворительно!
Введите оценку: 7 Нет такой оценки!
```

Если на всех ветвях убрать оператор break, то результат может выглядеть следующим образом:

```
Введите оценку: 3 Это удовлетворительно!
      Это хорошо!
      Это отлично!
      Нет такой оценки!
```

В этом случае выполнились операторы на всех ветвях, начиная с той, которая помечена константой 3.

Возможны задачи, в которых такой порядок выполнения ветвей оператора выбора может оказаться полезным. В следующем фрагменте программы происходит возведение вещественного числа x в целую степень n, где n изменяется в диапазоне от 1 до 5.

```
y = 1.0;
switch( n )
{
case 5:  y = y * x;
case 4:  y = y * x;
case 3:  y = y * x;
case 2:  y = y * x;
case 1:  y = y * x; cout << "y = " << y; break;
default: cout << "Степень больше 5";
}
```

Упражнения

1. Составить программу упорядочения по возрастанию значений в трех переменных.
2. Составить программу, которая выводит на экран меню:

    1. Первое
    2. Второе
    3. Третье

и в зависимости от выбранного пункта выдает одну из надписей:
«Получите суп», «Получите картошку», «Получите компот», «Оставайтесь голодным».
Написать два варианта программы: с использованием условного оператора if и с использованием переключателя.

3. Составить программу решения квадратного уравнения $ax^2 + bx + c = 0$, учитывающую все возможные варианты исходных данных:

    1) $a = 0,\ b = 0,\ c = 0$;
    2) $a = 0,\ b = 0,\ c \neq 0$;
    3) $a = 0,\ b \neq 0$;
    4) $a \neq 0,\ D \geq 0$ (D — дискриминант);
    5) $a \neq 0,\ D < 0$.

В каждом случае должно выводиться соответствующее решение или сообщение.

Литература

1. **Семакин И.Г., Шестаков А.П.** Основы программирования: Учебник.- М.: Мастерство, 2002.- 432 с.