Модуль 1. Основы программирования

Тема 1.6. Условные конструкции: if else, switch

2 часа

Оглавление

1.6. Условные конструкции: if else, switch

1.6.1. Конструкция if-else

Условный оператор if проверяет истинность выражения. Если проверка выражения покажет, что оно истинно (возвращается значение **true**), то выполняются действия после условия, в противном случае (возвращается значение **false**) выполняются команды после else.

Например,

```
If (x == 0);
{
        x=1;
}
else
{
        x=-1;
}
```

Условные операторы используются для создания точек ветвления в программе, когда тот или иной фрагмент кода выполняется в зависимости от того, выполняется или нет условие.

После ключевого слова if следует условие в круглых скобках. Условие может быть реализовано:

1) в полной форме:

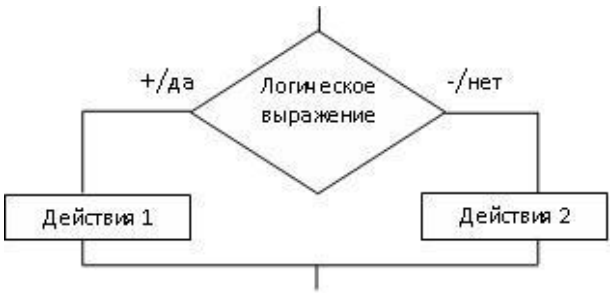
   if (условие) Действия 1

   else Действия 2

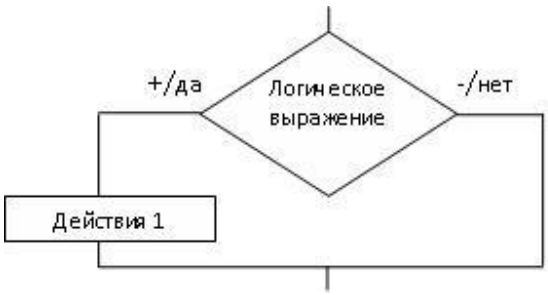
2) не в полной форме:

   if (условие) Действия 1

В первом случае при выполнении условия происходит выполнение Действий 1, при невыполнении условия - Действия 2. Во втором случае, при выполнении условия, происходит выполнение Действий 1, при невыполнении условия происходит выполнение следующей, после условного оператора, строки команд.



а) Блок-схема полной формы условного оператора



б) Блок-схема неполной формы условного оператора

Условные выражения

Выражение, проверяемое в условном операторе, может быть простым, а может быть составным. Например:

```
1. if (price < 100) //простое условие
2. if (x!=0) //простое условие
3. if (x>0 && x<1) //составное условие
4. if (s.equals("Russia") && g.equals("Ufa"))//составное условие
5. if (s.equals("Sweden") || s.equals("Finland") ||
   s.equals("Denmark"))  //составное условие
```

Условие выполнится если

1. значение переменной price будет меньше 100
2. значение переменной x будет не равно 0
3. значение переменной x будет находиться в интервале (0;1), те должны выполнится оба условия
4. значение строковых переменных s, g будут соответственно равны Russia и Ufa.
5. в строке s содержится название одной из трех стран Sweden, Finland, Denmark

	*Позже мы от примитивных типов Java перейдем к использованию объектов. При проверке их на равенство оператор == укажет нам лишь на то, являются ли объекты в памяти одними и теми же переменными. Если же необходимо проверить логическое равенство объектов, то следует использовать метод equals.*

Блоки

В ситуации, когда операторов более одного, они всегда заключаются в фигурные скобки и составляют блок операторов. Можно записать и пустой блок, просто пару фигурных скобок {}.

Блоки применяются, если:

- необходимо использовать несколько операторов вместо одного оператора;

Например, цена на услугу вычисляется с учетом инфляции

```
if (inf > 0){
  price+=inf*price/100;
  System.out.println("Цена: "+ price);
}
```

- для ограничения области действия переменных.

Результат работы следующего примера показывает что область видимости локальных переменных ограничена блоком, в котором они определены.

```
import java.io.PrintStream;
import java.util.Scanner;

public class MyProgram {
  public static Scanner in = new Scanner(System.in);
  public static PrintStream out = System.out;
  /*
   * Объявление и инициализация статической переменной v –скорость, она
   * известна во всех блоках
   */
  static int v = 15;

  public static void main(String[] args) {
      int p; // объявление переменной p
      System.out.print("Введите цифру: ");
      p = in.nextInt(); // p присваиваем введенное значение
      if (p == 1) {
   /*
    * инициализируем переменную v в блоке и выводим значение, в
    * случае выполнения условия
    */
          int v = 5;
          System.out.println("Пешеход. Скорость v =" + v + " км/ч");
      } else {
   /*
    * в случае невыполнения условия, инициализируем переменную v в
    * данном блоке и выводим значение
    */
          int v = 12;
          System.out.println("Велосипедист. Скорость v =" + v + " км/ч");
      }
      // в любом случае выводим значение v по умолчанию
      System.out.println("Предельная скорость на тротуаре: " + v + " км/ч");
  }
}
```

Проверим, что выведет программа, если условие будет истинно.

```
Введите цифру: 1
```

```
Пешеход. Скорость v =5 км/ч
Предельная скорость на тротуаре: 15 км/ч
```

Теперь посмотрим на работу программу, если условие ложно:

```
Введите цифру: 2
Велосипедист. Скорость v =12 км/ч
Предельная скорость на тротуаре: 15 км/ч
```

Зачастую блоки используют и для одного оператора с целью облегчения чтения программы и профилактики ошибок при внесении изменений в текст программы. Т.е., если в ветке реализуется только одна команда, то можно не записывать фигурные скобки:

```
if (x<y) //проверяем условие в скобках
      x = y - x; // выполняется, если условие=true
else
      x = x - y; // выполняется, если условие=false
```

А можно записать, выделив блок в фигурные скобки:

```
if (x < y)  {
    x = y - x;
}
else  {
    x = x - y;
}
```

Пример 1
Дано двузначное число. Определить входит ли в него цифра 3.

```
import java.io.PrintStream;
import java.util.Scanner;

public class MyProgram {
  public static Scanner in = new Scanner(System.in);
  public static PrintStream out = System.out;

  public static void main(String[] args) {
      int x1,x2; //объявили переменные x1 и x2
      x1 = in.nextInt(); // считали данные в x1
      x2 = x1%10;   //вторая цифра
      x1 = x1/10;  //первая цифра
      if (x2==3 || x1==3) {
          System.out.println("В числе присутствует цифра 3 ");
      }
      else {
          System.out.println("В числе нет цифры 3");
      }
      System.out.println( x1"; " + x2);
  }
}
```

Разберем условный оператор в этом примере. Найдем последнюю цифру двузначного числа x%10 (например, остаток числа 35 при делении на 10 равен 5). Если эта цифра равна 3 или первая цифра двузначного числа в результате целочисленного деления x/10 равна 3 (в том же примере 35/10=3), то выведем сообщение "В числе присутствует цифра 3 ". Если ни одно из условий не выполнится, то выведем сообщение "В числе нет цифры 3".

Вспомним изученную тернарную операцию (?:) - это по сути форма условного оператора. Сначала проверяется логическое выражение. Если выражение истинно, то результат всей операции - значение стоящее сразу после вопросительного знака, в противном случае – значение стоящего после двоеточия

```
x = y > 0 ? 10 : -10;
```

аналог

```
if (y > 0) {
   x = 10;
}
else {
   x = -10;
}
```

Примером реализации тернарной операции может быть задача с делением на 0. Если значение переменной n равно 0, то ей присваивается значение переменной m, а иначе n = m / n.

```
n == 0 ? m : m / n
```

Конструкция из вложенных условных операторов

Условный оператор if содержащий несколько else if блоков имеет следующий синтаксис:

```
if (условное выражение1) {блок1 команд }

else if (условие2){блок2 команд }

else if (условие3){блок3 команд }

…

else if (условиеN){блокN команд }

else { блокN+1 команд }
```

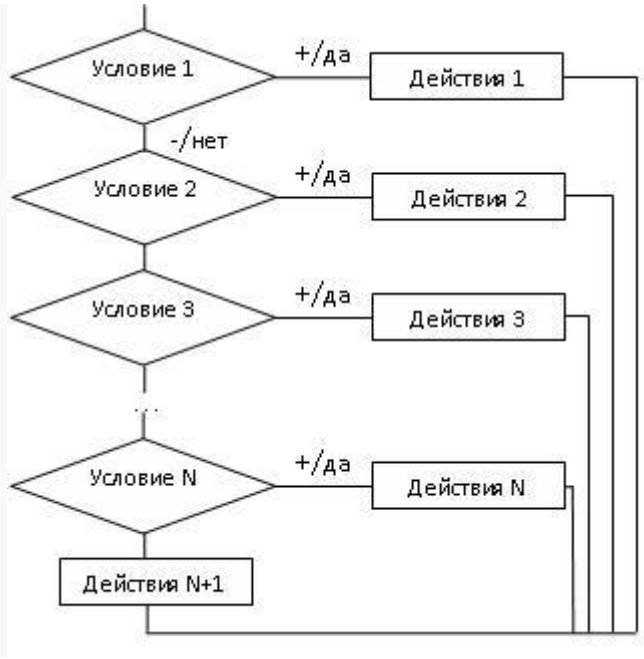


Рис.3. Блок-схема конструкции из вложенных условных операторов

Пример 2

Дано двузначное число. Определить какая из его цифр больше первая или вторая.

Помимо задания дополним нашу программу еще проверкой на то, что введенное пользователем число является действительно двузначным - такие проверки позволяют сделать программу более устойчивой. В иоге получим программу с несколькими вложенными конструкциями If else:

```
import java.io.PrintStream;
import java.util.Scanner;

public class MyProgram {
  public static Scanner in = new Scanner(System.in);
  public static PrintStream out = System.out;

  public static void main(String[] args) {
      int x1, x2;
      x1 = in.nextInt();
      if (x1>9 && x1<100) { //Число двузначное?
          x2 = x1%10;   //вторая цифра
          x1 = x1/10;  //первая цифра
          System.out.println(x1 + "; " + x2);
          if (x1  == x2) {
              System.out.println("В числе одинаковые цифры");
          } else if (x2 < x1 ) {
              System.out.println("Первая цифра больше ");
          } else {
              System.out.println("Вторая цифра больше ");
          }
      }
      else       //Число не двузначное
          System.out.println("Введенное число не двузначное");
  }
}
```

1.6.2. Конструкция switch-case

Оператор выбора **switch-case** позволяет сделать выбор между несколькими вариантами.

```
Switch(выражение) {
case значение1: команды1; break;
case значение2: команды2; break;
…
case значениеN: командыN; break;
default: командыN+1;
}
```

Оператор **switch-case** работает так:

1. вычисляется значение выражения;
2. если такое значение найдено, то дальнейшая проверка не производится, а выполняются команды, соответствующие выбранной ветви, после чего управление передается оператору, следующему за фигурной скобкой, которая закрывает всю конструкцию switch-case. Каждый case блок заканчивается командой **break**, для прерывания оператора switch-case;
3. если подходящего значения в перечне нет, то выполняются команды, стоящие после ключевого слова default. Если ветви **default** нет, то не выполняется ничего. В конструкции switch может быть применен только один оператор default.

В качестве выражения switch может быть использована переменная типа **byte, short, int, char**, но не **long**. Кроме простых целых типов допускаются их классы-оболочки, перечисления и строки символов типа String. При этом тип значений должен соответствовать типу выражения.

В операторе **switch не может быть двух case** с одинаковыми значениями. Поэтому нижеприведенная конструкция недопустима.

```
switch(x){
      case 1:
            System.out.println("One ");
            break;
      case 1:
            System.out.println("X=1");
            break;
      case 3:
            System.out.println("X>2");
}
```

Также важно учитывать, что после case следует именно значение, а не условие! Поэтому следующая конструкция вызовет ошибку.

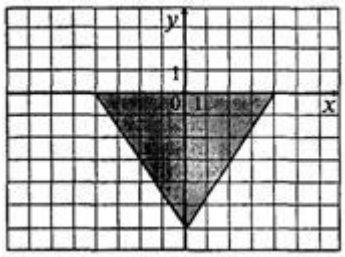
```
switch(x){
      case x>0:
            System.out.println("Число положительное ");
            break;
      case x<0:
            System.out.println("Число отрицательное");
            break;
      default:
            System.out.println("Ноль");
      }
```

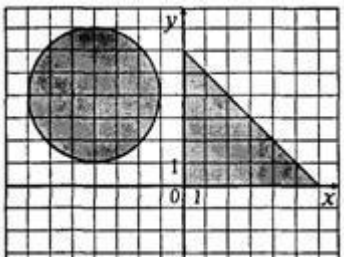
Пример программы на применение оператора множественного выбора:

```
import java.io.PrintStream;
import java.util.Scanner;
/* Небольшие фрагменты кода, иллюстрирующие использование
 * оператора switch
 */
public class MyProgram {
   public static Scanner in = new Scanner(System.in);
   public static PrintStream out = System.out;
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      System.out.println("Введите:\n1 - работа с переменной целого типа\n2 -
работа с переменной вещественного типа");
      int menu = in.nextInt();// считываем целое число
      switch (menu) {
          case 1:
              System.out.println("Введите целое число:");
              int a = in.nextInt();// считываем целое число a
              // Определяем является ли число a четным
              if ((a % 2) == 0)
                   System.out.println("Число " + a + " - четное");
              else
                   System.out.println("Число " + a + "- нечетное");
              break;
          case 2:
              double b = 24.567;
              // Определяем дробную часть
              System.out.println("Дробная часть числа " + b + "= "
                             + (b - (int) b));
              break;
          default:
              System.out.println("Вы ввели не 1 и не 2");
      }
      System.out.println("Конец!");
   }
}
```

Задание 1.6.1

Запишите условия для точек, принадлежащих выделенной области

А) 

Б) 

Задание 1.6.2

1) Написать собственный пример на нахождение максимума, минимума среди нескольких введенных переменных с использованием оператора ветвления.

2) Вывести на экран номер четверти координатной плоскости, которой принадлежит точка с координатами (x,y), при условии что x≠0 и y≠0.

Задание 1.6.3

1) Написать программу, которая по введенному номеру единицы измерения (1-дециметр, 2-километр, 3-метр, 4-миллиметр, 5-сантиметр, 6-дюймы, 7-футы, 8-мили) и длине отрезка выдавала бы соответствующее значение длины отрезка в метрах.

2) Для заданного 0<n<130 , рассматриваемого как возраст человека, вывести фразу вида «Вам 21 год», «Вам 32 года», « Вам 12 лет».

Задание 1.6.4

Для того, чтобы попрактиковаться в решении задач с использованием условных конструкций попробуйте решить задачи http://informatics.msk.ru №№ 294, 1459, 2961

Благодарности

Компания Samsung Electronics выражает благодарность за участие в подготовке данного материала преподавателю IT ШКОЛЫ SAMSUNG Зайдуллиной Светлане Галимулловне.