Условные операторы

В этой теме вы узнали, что такое условные операторы и как с ними работать. Вот что нужно запомнить:

Ввод данных через консоль

Чтобы программа могла считать данные из консоли, нужно выполнить такие действия:

1. Импортировать Scanner из пакета `java.util`.

```
import java.util.Scanner;  // Такая строка должна появиться перед объявлением класса
```

Если нужный тип данных хранится где-то вне программы (тип `Scanner` хранится в пакете `java.util`), то для того, чтобы его использовать, его сначала нужно оттуда **импортировать**.

2. Внутри программы, в методе `main()`, объявить переменную `scanner` с типом `Scanner` и присвоить ей такое значение `new Scanner(System.in)`.

```
Scanner scanner = new Scanner(System.in); // Объявили переменную с типом Scanner
```

`scanner` умеет считывать данные из потока ввода `System.in` и относится к сложным типам переменных.

3. Последний шаг — создать переменную нужного типа и записывать в неё результат ввода данных. Это происходит с помощью метода для считывания информации из консоли, например, метод `scanner.nextLine();` позволяет считать строки.

```
String command = scanner.nextLine(); // В command будет занесён ввод строки из консоли
```

С помощью `scanner` можно считать такую информацию:

`nextLine()` — строки.

`nextInt()` — целые числа.

`nextDouble()` — числа с дробной частью.

Конструкция `if(){}`

Для создания конструкции с условием или **ветвления** в программе используется условный оператор `if`.

```
if (условие) {
  // Код #1 выполнится, только если условие == true
}
// Код #2 выполнится в любом случае
```

Ветвление можно прочитать так: «**ЕСЛИ** условие истинно, **ТО** выполнить код #1».

Обратите внимание, код #1 сгруппирован в фигурных скобках `{}` — такие участки программы называют блоками кода. Если внутри блока объявлены переменные, то они не видны за его пределами. Это называется **областью видимости переменных**. Например:

```
boolean isVisible = true;
if(isVisible) { // ветвление с if
  String condition = "истину"; // объявили переменную внутри блока кода
  System.out.println("Я наконец увидел" + condition); // здесь она видна
}
```

```
// Вне блока ветвления переменная condition не видна
System.out.println("Я наконец увидел" + condition);  // Здесь произойдет ошибка
```

Если вы попытаетесь запустить такой код, то произойдёт ошибка, потому что переменная `condition` видна только внутри блока кода ветвления `if` и не видна снаружи, но вы попытались к ней обратиться.

Конструкция `if(){} else{}`

Более сложный вариант конструкции `if(){}` ветвление `if(){} else{}`:

```
if (условие) {
   // Код #1 выполнится, только если условие == true
} else {
  // Код #2 выполнится, только если условие == false
}
// Код #3 выполнится в любом случае
```

Этот можно прочитать так: «**ЕСЛИ** условие истинно, **ТО** выполнить код #1 внутри блока `if` , **ИНАЧЕ** выполнить код #2 внутри блока `else`».

Конструкцию `if(){} else{}` можно усложнять:

```
if (условие 1) {
  код #1
} else if (условие 2) {
  код #2
} else {
  код #3
}
код #4
```

В этом случае ветвление читается так.

- Если `условие 1` истинно, тогда выполнится код #1.
- Когда `условие 1` ложно, но `условие 2` истинно, сработает код #2.
- Если оба условия ложны, сработает код #3.

Вложенные условия

Добавить дополнительное условие в уже существующее ветвление можно также с помощью вложенного условия.

```
if (условие 1) {

  if (условие 2) { // Вложенное условие if
    //код #1
  } else {
    //код #2
  }
} else {
  //код #3
}
```

Читается так: «**ЕСЛИ** условие 1 истинно, **ТО** *проверить условие 2,* **ИНАЧЕ** выполнить код #3.

Проверка условия 2. **ЕСЛИ** условие 2 истинно, **ТО** выполнить код #1 внутри блока `if`, **ИНАЧЕ** выполнить код #2 внутри блока `else` .

Усложненные ветвления и ветвления с вложенными условиями — это разные конструкции.

Внутри вложенного `if` можно написать сколько угодно ещё вложенных условных выражений. Но старайтесь не увлекаться вложенностью — ваш код должен оставаться доступным для понимания.

Тип boolean

Логический тип данных `boolean` применяют в условных выражениях для проверки выполнения условия.

Переменная типа `boolean` может быть в одном из двух состояний — `true` или `false`. Если на какой-то вопрос можно ответить «да», то это `true`, если «нет» — `false`.

Значение типа `boolean` появляется при сравнении чисел, строк или значений переменных:

```
boolean isMore = 3 > 4;// результат сравнения запишется в переменную isMore
```

Варианты операторов сравнения:

`>` — строго больше

`<` — строго меньше

`>=` — больше либо равно

`<=` — меньше либо равно

`!=` — не равно

`==` — равно

Для сравнения между собой строк `String` эти операторы не работают, существует особый метод `equals` (англ. «сравнивать»). Его синтаксис выглядит так:

```
String str1 = "Какой-то текст";
String str2 = "Какой-то " + "текст";

boolean isSameString = str1.equals(str2); // Переменная isSameString = true
isSameString = str1.equals("Другая строка"); // Теперь переменная isSameString = false
```

Слово `equals` сравнивает строку `str1` с той, что написана в круглых скобках — `str2`. Если они равны, то будет возвращено `true`, иначе — `false`. Можно сравнивать не только переменные с типом `String`, но и текстовые строки в кавычках.

Название `boolean` (булевой) переменной принято начинать с префикса `is`. Например: `isEven`, `isNegative`.

Конструкция `if(){} else{}`

Более сложный вариант конструкции `if(){}` ветвление `if(){} else{}`:

```
if (условие) {
   // Код #1 выполнится, только если условие == true
} else {
  // Код #2 выполнится, только если условие == false
```

```
}
// Код #3 выполнится в любом случае
```

Этот можно прочитать так: «**ЕСЛИ** условие истинно, **ТО** выполнить код #1 внутри блока `if` , **ИНАЧЕ** выполнить код #2 внутри блока `else`».

Конструкцию `if(){} else{}` можно усложнять:

```
if (условие 1) {
  код #1
} else if (условие 2) {
  код #2
} else {
  код #3
}
код #4
```

В этом случае ветвление читается так.

- Если `условие 1` истинно, тогда выполнится код #1.
- Когда `условие 1` ложно, но `условие 2` истинно, сработает код #2.
- Если оба условия ложны, сработает код #3.

Вложенные условия

Добавить дополнительное условие в уже существующее ветвление можно также с помощью вложенного условия.

```
if (условие 1) {

  if (условие 2) { // Вложенное условие if
    //код #1
  } else {
    //код #2
  }
} else {
  //код #3
}
```

Читается так: «**ЕСЛИ** условие 1 истинно, **ТО** *проверить условие 2,* **ИНАЧЕ** выполнить код #3.

Проверка условия 2. **ЕСЛИ** условие 2 истинно, **ТО** выполнить код #1 внутри блока `if`, **ИНАЧЕ** выполнить код #2 внутри блока `else` .

Усложненные ветвления и ветвления с вложенными условиями — это разные конструкции.

> 💡 Внутри вложенного `if` можно написать сколько угодно ещё вложенных условных выражений. Но старайтесь не увлекаться вложенностью — ваш код должен оставаться доступным для понимания.

Тип boolean

Логический тип данных `boolean` применяют в условных выражениях для проверки выполнения условия.

Переменная типа `boolean` может быть в одном из двух состояний — `true` или `false`. Если на какой-то вопрос можно ответить «да», то это `true`, если «нет» — `false`.

Значение типа `boolean` появляется при сравнении чисел, строк или значений переменных:

```
boolean isMore = 3 > 4;// результат сравнения запишется в переменную isMore
```

Варианты операторов сравнения:

`>` — строго больше

`<` — строго меньше

`>=` — больше либо равно

`<=` — меньше либо равно

`!=` — не равно

`==` — равно

Для сравнения между собой строк `String` эти операторы не работают, существует особый метод `equals` (англ. «сравнивать»). Его синтаксис выглядит так:

```
String str1 = "Какой-то текст";
String str2 = "Какой-то " + "текст";

boolean isSameString = str1.equals(str2); // Переменная isSameString = true
isSameString = str1.equals("Другая строка"); // Теперь переменная isSameString = false
```

Слово `equals` сравнивает строку `str1` с той, что написана в круглых скобках — `str2`. Если они равны, то будет возвращено `true`, иначе — `false`. Можно сравнивать не только переменные с типом `String`, но и текстовые строки в кавычках.

> 💡 Название `boolean` (булевой) переменной принято начинать с префикса `is`. Например: `isEven`, `isNegative`.

На этом все! Отдохните и подготовьтесь к следующей теме: "Циклы".