Задания из третьей Главы учебника

**Упражнение 1.** Цель: Создание Java-программы, которая демонстрирует использование условных операторов if в Java для проверки различных условий. Она выводит соответствующие сообщения на основе выполнения условий.

## If.java

```
class If {
    public static void main(String[] args) {
        // Проверяем, что 5 больше чем 1, и выводим сообщение, если условие выполняется.
        if (5 > 1)
            System.out.println("Пять больше чем один.");

        // Проверяем, что 2 меньше чем 4.
        if (2 < 4) {
            // Если условие выполняется, выводим два сообщения.
            System.out.println("Два меньше четырех.");
            System.out.println("Проверка выполнена успешно.");
        }

        int num = 8;

        // Проверяем, что число num находится в диапазоне от 6 до 9 включительно или равно
12.
        if (((num > 5) && (num < 10)) || (num == 12))
            System.out.println("Число в диапазоне от 6 до 9 включительно или равно 12");
    }
}
```

Результат

```
Пять больше чем один.
Два меньше четырех.
Проверка выполнена успешно.
Число в диапазоне от 6 до 9 включительно или равно 12
```

Объяснение кода:
Приведенный код представляет собой пример использования оператора **if** в языке программирования Java для выполнения различных условных проверок. Давайте разберем его по частям:

1. **if (5 > 1) System.out.println("Пять больше чем один.");**
    - Здесь создается условное выражение. Если 5 больше чем 1, то выполняется команда **System.out.println("Пять больше чем один.");**, которая выводит текст "Пять больше чем один." в консоль. Так как это условие верно, сообщение будет выведено.
2. **if (2 < 4) { ... }**
    - Здесь создается еще одно условное выражение. Если 2 меньше чем 4, то выполняются команды внутри блока кода в фигурных скобках. В данном случае, выводятся два сообщения в консоль: "Два меньше четырех." и "Проверка выполнена успешно."
3. **int num = 8;**

- Создается целочисленная переменная **num**, которая получает значение 8.

4. **if (((num > 5) && (num < 10)) || (num == 12)) { ... }**
    - Здесь создается более сложное условное выражение. Оно проверяет два условия:
        - **(num > 5) && (num < 10)** - Если число **num** больше 5 и меньше 10, то первое условие верно.
        - **(num == 12)** - Если число **num** равно 12, то второе условие верно.
    - Оператор **||** означает "или". Таким образом, если хотя бы одно из этих условий верно, то внутренний блок кода выполняется. В данном случае, если число **num** равно 8, что удовлетворяет первому условию, то программа выводит "Число в диапазоне от 6 до 9 включительно или равно 12."

Итак, код демонстрирует различные способы использования оператора **if** для проверки условий и выполнения кода на основе этих условий.

**Упражнение 2.** Цель: Написать программу, которая на основе значения часов выводит приветствие в зависимости от времени суток.

**Else.java**

```
import java.util.Scanner;

public class Else {
    public static void main(String[] args) {
        // Создаем объект Scanner для считывания ввода с консоли
        Scanner scanner = new Scanner(System.in);

        // Спрашиваем пользователя ввести значение часов (hrs)
        System.out.print("Введите часы (hrs): ");

        // Считываем введенное значение и сохраняем его в переменной hrs
        int hrs = scanner.nextInt();

        // Закрываем объект Scanner, так как он больше не нужен
        scanner.close();

        // Выполняем условные проверки и выводим соответствующее приветствие
        if (hrs < 13) {
            System.out.println("Доброе утро: " + hrs);
        } else if (hrs < 18) {
            System.out.println("Добрый день: " + hrs);
        } else {
            System.out.println("Добрый вечер: " + hrs);
        }
    }
}
```

* Код отличается от представленной в учебнике. Потому что, я добавил возможность из консоля задавать значение переменного **hrs** чтобы, сэкономить время и ресурсы компьютера, постоянно не редактировать и перекомпилировать код.

Результат

```
Введите часы (hrs): 11
Доброе утро: 11

Введите часы (hrs): 15
Добрый день: 15

Введите часы (hrs): 21
Добрый вечер: 21
```

Объяснение кода:

1. Мы создаем объект **Scanner**, чтобы считывать ввод пользователя с консоли.
2. Затем программа просит пользователя ввести значение часов (переменная **hrs**) с помощью **System.out.print**.
3. Введенное значение сохраняется в переменной **hrs**.
4. Мы закрываем объект **Scanner**, так как он больше не нужен.
5. Далее программа выполняет условные проверки:
    - Если **hrs** меньше 13, выводится "Доброе утро: " и значение **hrs**.
    - Иначе, если **hrs** меньше 18, выводится "Добрый день: " и значение **hrs**.
    - В противном случае (если **hrs** не соответствует предыдущим условиям), выводится "Добрый вечер: " и значение **hrs**.

**Упражнение 3.** Цель: Написать программу, использующую оператор **switch** и условные выражения для определения количества дней в указанном месяце и вывода результата.

Switch.java

```
class Switch {
    public static void main(String[] args) {
        int month = 2, year = 2016, num = 31;
        // Используем оператор switch для выбора количества дней в месяце
        switch (month) {
            case 4: case 6: case 9: case 11:
            // Если месяц - апрель, июнь, сентябрь или ноябрь, устанавливаем num в 30
                num = 30;
                break;
            case 2:
            // Если месяц - февраль, используем условный оператор для определения
високосного года
                num = (year % 4 == 0) ? 29 : 28;
                break;
        }
        // Выводим результат в формате "месяц/год: количество дней"
        System.out.println(month + "/" + year + ": " + num + " дней");
    }
}
```

Результат

```
2/2016: 29 дней
```

Объяснение кода:

1. Создаем переменные **month**, **year** и **num** и инициализируем их значениями: **month** установлен в 2 (представляет февраль), **year** установлен в 2016, и **num** установлен в 31.

2. Затем используется оператор **switch**, чтобы определить количество дней в месяце, основываясь на значении переменной **month**.

3. В блоке **switch**, у нас есть два **case** (случая):

    - Если **month** соответствует одному из значений: 4, 6, 9 или 11 (месяцы с 30 днями), мы устанавливаем **num** в 30 дней, так как эти месяцы имеют 30 дней.

    - Если **month** равен 2 (февраль), мы используем условное выражение, чтобы проверить, является ли год високосным. Если год делится на 4 без остатка (**year % 4 == 0**), то **num** устанавливается в 29 (февраль в високосном году), в противном случае **num** устанавливается в 28 (февраль в невисокосном году).

4. Затем результат (количество дней в месяце) выводится на экран с использованием команды **System.out.println()**, в формате "месяц/год: количество дней". В данном случае, вывод будет "2/2016: 29 дней".

**Упражнение 4.** Цель: Написать программу на Java, используя вложенные циклы, для демонстрации работы с двумя уровнями итераций и подсчета общего числа итераций.

For.java

```
class For {
    public static void main(String[] args) {
        int num = 0;
        // Внешний цикл - итерируется по переменной i от 1 до 3
        for (int i = 1; i < 4; i++) {
            System.out.println("Внешний цикл i=" + i);
            // Внутренний цикл - итерируется по переменной j от 1 до 3
            for (int j = 1; j < 4; j++) {
                System.out.print("\tВнутренний цикл j=" + j);
                // Увеличиваем значение переменной num на 1 и выводим обновленное значение
                System.out.println("\t\tВсего num=" + (++num));
            }
        }
    }
}
```

Результат

```
Внешний цикл i=1
        Внутренний цикл j=1             Всего num=1
        Внутренний цикл j=2             Всего num=2
        Внутренний цикл j=3             Всего num=3
Внешний цикл i=2
        Внутренний цикл j=1             Всего num=4
        Внутренний цикл j=2             Всего num=5
        Внутренний цикл j=3             Всего num=6
Внешний цикл i=3
        Внутренний цикл j=1             Всего num=7
        Внутренний цикл j=2             Всего num=8
        Внутренний цикл j=3             Всего num=9
```

Объяснение кода:

1. Создаем переменную **num** и инициализируем её значением 0. Эта переменная будет использоваться для подсчета общего количества итераций.

2. Затем у нас есть внешний цикл **for**, который итерируется по переменной **i** от 1 до 3 включительно (**for (int i = 1; i < 4; i++)**).

    - На каждой итерации внешнего цикла выводится строка, содержащая значение **i**, например, "Внешний цикл i=1".

3. Внутри внешнего цикла у нас есть вложенный цикл **for**, который итерируется по переменной **j** от 1 до 3 включительно (**for (int j = 1; j < 4; j++)**).

    - На каждой итерации вложенного цикла выводится строка, содержащая значение **j**, например, "Внутренний цикл j=1".

4. Внутри внутреннего цикла мы выполняем следующее:

    - Мы выводим строку с табуляцией, чтобы создать отступ (например, "\tВнутренний цикл j=1").

    - Затем мы увеличиваем значение переменной **num** на 1 с помощью оператора **++**. Это увеличивает **num** на 1 и выводит его обновленное значение.

    - Выводится строка, содержащая значение **num**, например, "Всего num=1".

5. Этот процесс повторяется для всех возможных комбинаций значений **i** и **j**, что приводит к выводу всех итераций как внешнего, так и внутреннего циклов.

Таким образом, код иллюстрирует, как вложенные циклы могут использоваться для создания итераций в двумерной структуре, и как переменная **num** служит для подсчета общего числа итераций.

**Упражнение 5.** Цель: Написать программу на Java, используя цикл **while**, для создания обратного отсчета от числа 100 до 0 с шагом 10.

**While.java**

```java
class While {
    public static void main(String[] args) {
```

```
        int num = 100;
        // Используем цикл while для создания обратного отсчета
        while (num > 0) {
            // Выводим сообщение о текущем значении num
            System.out.println("Обратный отсчет с использованием While: " + num);
            // Уменьшаем значение num на 10 на каждой итерации
            num -= 10;
        }
    }
}
```

Результат

```
Обратный отсчет с использованием While: 100
Обратный отсчет с использованием While: 90
Обратный отсчет с использованием While: 80
Обратный отсчет с использованием While: 70
Обратный отсчет с использованием While: 60
Обратный отсчет с использованием While: 50
Обратный отсчет с использованием While: 40
Обратный отсчет с использованием While: 30
Обратный отсчет с использованием While: 20
Обратный отсчет с использованием While: 10
```

Объяснение кода:

1. **int num = 100;**: Создается переменная **num** и инициализируется значением 100. Это начальное значение, с которого начнется обратный отсчет.

2. **while (num > 0) {**: Это начало блока цикла **while**. Условие **num > 0** проверяется на каждой итерации. Цикл будет выполняться, пока значение **num** больше 0.

3. **System.out.println("Обратный отсчет с использованием While: " + num);**: На каждой итерации цикла выводится сообщение на консоль, которое включает текущее значение **num**. Например, если **num** равно 100, то сообщение будет "Обратный отсчет с использованием While: 100".

4. **num -= 10;**: Здесь значение переменной **num** уменьшается на 10 на каждой итерации. Это отвечает за шаг обратного отсчета.

5. Завершение блока цикла **while**: Когда значение **num** становится меньше или равно 0, выполнение цикла прекращается, и программа продолжает выполнение далее.

**Упражнение 6.** Цель: Написать программу, использующую цикл **do-while**, для вывода значений переменной **num** и модификации этой переменной до тех пор, пока **num** меньше нуля.

DoWhile.java

```
class DoWhile {
    public static void main(String[] args) {
        int num = 100;
        // Используем цикл do-while для вывода текущего значения переменной num, пока оно
меньше нуля.
        do {
```

```
            System.out.println("Используем DoWhile: " + num);
            // Модифицируем значение переменной num на каждой итерации, увеличивая его на
10, чтобы избежать бесконечного цикла.
            num += 10;
        } while (num < 0); // Условие проверяет, меньше ли значение num нуля.
    }
}
```

Результат

```
Используем DoWhile: 100
```

Объяснение кода:

1. **int num = 100;** - Здесь создается целочисленная переменная **num** и инициализируется значением 100.
2. **do { ... } while (num < 0);** - Это конструкция цикла **do-while**. Она гарантирует выполнение блока кода, заключенного в фигурные скобки, по меньшей мере один раз, даже если условие в скобках **while** ложно. В данном случае, цикл будет выполняться до тех пор, пока **num** меньше нуля.
3. **System.out.println("Используем DoWhile: " + num);** - Этот оператор выводит на консоль текущее значение переменной **num**, добавляя текст "Используем DoWhile: ".
4. **num += 10;** - Эта строка увеличивает значение переменной **num** на 10 на каждой итерации цикла. Это сделано для того, чтобы постепенно увеличивать значение **num** и, в конечном итоге, выйти из цикла.
5. **while (num < 0);** - Это условие проверяет, меньше ли значение **num** нуля. Если оно меньше нуля, цикл продолжает выполняться.

Таким образом, программа будет выводить текущее значение **num** и увеличивать его на 10 на каждой итерации до тех пор, пока **num** останется положительным числом. Как только **num** становится неположительным (меньше или равно нулю), цикл завершается.

**Упражнение 7.** Цель: Написать программу, демонстрирующую использование операторов **break** и **continue** для управления выполнением вложенных циклов **for**.

**Break.java**

| Изменения в коде | Результаты |
|---|---|
| `class Break {`<br>`  public static void main(String[] args) {`<br>`  for (int i = 1; i < 4; i++) {`<br>`  for (int j = 1; j < 4; j++) {`<br>`  System.out.println("Итерация i=" + i + " j=" + j);`<br>`  }`<br>`  }`<br>`  }`<br>`}` | `Итерация i=1 j=1`<br>`Итерация i=1 j=2`<br>`Итерация i=1 j=3`<br>`Итерация i=2 j=1`<br>`Итерация i=2 j=2`<br>`Итерация i=2 j=3`<br>`Итерация i=3 j=1`<br>`Итерация i=3 j=2`<br>`Итерация i=3 j=3` |

**Объяснение кода:**
Код представляет собой вложенные циклы **for**, которые выполняют итерации по значениям **i** и **j** в пределах от 1 до 3. На каждой итерации внутреннего цикла выводится строка, содержащая значения **i** и **j**. Таким образом, программа выведет все возможные комбинации пар **(i, j)** от 1 до 3.

**Изменить код:**
Добавьте оператор **break** в начало блока операторов внутреннего цикла, чтобы выполнить выход из этого цикла, затем перекомпимлируйте и запустите заново программу

```
class Break {
    public static void main(String[] args) {
        for (int i = 1; i < 4; i++) {
            for (int j = 1; j < 4; j++) {
                if (i == 2 && j == 1) {
                    System.out.println("Выход из внутреннего цикла при
i=" + i + " j=" + j);
                    break; // Оператор break для выхода из внутреннего
цикла.
                }
                System.out.println("Итерация i=" + i + " j=" + j);
            }
        }
    }
}
```

```
Итерация i=1 j=1
Итерация i=1 j=2
Итерация i=1 j=3
Выход из внутреннего
цикла при i=2 j=1
Итерация i=3 j=1
Итерация i=3 j=2
Итерация i=3 j=3
```

**Объяснение кода:**
Код представляет собой вложенные циклы **for**, где внутренний цикл выполняется внутри внешнего цикла. Происходит итерация по **i** и **j** от 1 до 3 в обоих циклах, что создает девять комбинаций итераций. Внутри циклов происходит вывод текста, отображающего текущие значения **i** и **j** для каждой итерации.
Однако, при условии, что **i** равно 2 и **j** равно 1, используется оператор **break**. Оператор **break** прерывает выполнение внутреннего цикла, при этом отображается сообщение "Выход из внутреннего цикла при i=2 j=1". Это предотвращает выполнение оставшихся итераций внутреннего цикла.
Для всех остальных комбинаций значений **i** и **j**, выполняются обычные итерации внутреннего цикла, и отображается сообщение "Итерация i=X j=Y", где **X** и **Y** - текущие значения **i** и **j**.

**Изменить код:**
Добавьте оператор continue в начало блока операторов внутреннего цикла, чтобы пропустить первую итерацию внутреннего цикла, затем перекомпилируйте и запустите заново программу

```
class Break {
    public static void main(String[] args) {
        for (int i = 1; i < 4; i++) {
            for (int j = 1; j < 4; j++) {
                if (i == 2 && j == 1) {
                    System.out.println("Выход из внутреннего цикла при
i=" + i + " j=" + j);
                    break; // Оператор break для выхода из внутреннего
цикла.
                }
                if (i == 1 && j == 1) {
                    System.out.println("Продолжение работы внутреннего
цикла при i=" + i + " j=" + j);
                    continue; // Оператор continue для пропуска первой
итерации внутреннего цикла.
                }
                System.out.println("Итерация i=" + i + " j=" + j);
            }
        }
    }
}
```

```
Продолжение работы
внутреннего цикла при
i=1 j=1
Итерация i=1 j=2
Итерация i=1 j=3
Выход из внутреннего
цикла при i=2 j=1
Итерация i=3 j=1
Итерация i=3 j=2
Итерация i=3 j=3
```

**Объяснение кода:**
Этот код использует операторы **break** и **continue** для контроля над выполнением вложенных циклов **for**. Оператор **break** позволяет выйти из внутреннего цикла при определенных условиях, а оператор **continue** пропускает текущую итерацию внутреннего цикла и переходит к следующей. Это демонстрируется в коде при помощи условных проверок на значения **i** и **j**.

**Упражнение 8.** Создание программы, которая демонстрирует использование операторов **break** и **continue** для управления выполнением вложенных циклов.

**Bitwise.java**

| Изменения в коде | Результаты |
|---|---|

```
1.  class Label {
2.      public static void main(String[] args) {
3.          for (int i = 1; i < 4; i++) {
4.              for (int j = 1; j < 4; j++) {
5.                  System.out.println("Итерация i=" + i + " j=" + j);
                }
6.          }
7.      }
8.  }
9.
```

```
1. Итерация i=1 j=1
2. Итерация i=1 j=2
3. Итерация i=1 j=3
4. Итерация i=2 j=1
5. Итерация i=2 j=2
6. Итерация i=2 j=3
7. Итерация i=3 j=1
8. Итерация i=3 j=2
9. Итерация i=3 j=3
```

**Объяснение кода:**

Код представляет программу с двумя вложенными циклами **for**, выполняющими итерации в переменных **i** и **j** от 1 до 3. На каждой итерации выводится сообщение с текущими значениями **i** и **j**. Результат - 9 итераций в общей сложности.

**Изменить код:**

1. Отредактируйте начало внешнего цикла, пометив его меткой **outerLoop**.
Чтобы явно указать программе, что нужно перейти к этому внешнему циклу, поставьте имя данной метки после ключевого слова **continue**.
2. Добавьте оператор **continue** в начало блока операторов внутреннего цикла, чтобы перейти к следующей итерации внешнего цикла, затем перекомпилируйте и запустите заново программу.

```
1.  class Label {
2.      public static void main(String[] args) {
3.          outerLoop: for (int i = 1; i < 4; i++) {
4.              for (int j = 1; j < 4; j++) {
5.                  // Проверяем условие: если i равно 1 и j равно 1
6.                  if (i == 1 && j == 1) {
7.                      // Выводим сообщение и переходим к следующей
    итерации внешнего цикла
8.                      System.out.println("outerLoop продолжает
    работу при i=" + i + " j=" + j);
9.                      continue outerLoop;
10.                 }
11.                 // Выводим информацию о текущей итерации
12.                 System.out.println("Итерация i=" + i + " j=" + j);
                }
13.         }
14.     }
15. }
```

```
1. outerLoop
   продолжает работу
   при i=1 j=1
2. Итерация i=2 j=1
3. Итерация i=2 j=2
4. Итерация i=2 j=3
5. Итерация i=3 j=1
6. Итерация i=3 j=2
7. Итерация i=3 j=3
```

**Объяснение кода:**

Этот код представляет собой Java-программу, в которой есть два вложенных цикла **for**. Внутри внешнего цикла **for** есть условная проверка: если **i** равно 1 и **j** равно 1, то программа выводит сообщение "outerLoop продолжает работу при i=1 j=1" и переходит к следующей итерации внешнего цикла. В противном случае, программа выводит информацию о текущей итерации с помощью сообщения "Итерация i=" и "j=".

**Изменить код:**
Добавьте оператор **break** в начало блока операторов внутреннего цикла для выхода из внешнего цикла, затем снова скомпилируйте и запустите программу.

```
1.  class Label {
2.      public static void main(String[] args) {
3.          outerLoop: for (int i = 1; i < 4; i++) {
4.              for (int j = 1; j < 4; j++) {
5.                  // Проверяем условие: если i равно 1 и j равно 1
6.                  if (i == 1 && j == 1) {
7.                      // Выводим сообщение и переходим к следующей
    итерации внешнего цикла
8.                      System.out.println("outerLoop продолжает
    работу при i=" + i + " j=" + j);
9.                      continue outerLoop;
10.                 }
11.                 // Проверяем условие: если i равно 2 и j равно 3
12.                 if (i == 2 && j == 3) {
13.                     // Выводим сообщение и выходим из внешнего
    цикла
14.                     System.out.println("Выход из outerLoop при i="
    + i + " j=" + j);
15.                     break outerLoop;
16.                 }
17.                 // Выводим информацию о текущей итерации
18.                 System.out.println("Итерация i=" + i + " j=" + j);
19.             }
20.         }
21.     }
22. }
```

```
1. outerLoop
   продолжает работу
   при i=1 j=1
2. Итерация i=2 j=1
3. Итерация i=2 j=2
4. Выход из
   outerLoop при i=2
   j=3
```

**Объяснение кода:**
Код использует операторы **break** и **continue** для управления выполнением вложенных циклов **for**.

- Оператор **break** позволяет завершить внутренний цикл, когда определенное условие выполняется. В данном случае, если **i** равно 2 и **j** равно 3, внешний цикл завершается.
- Оператор **continue** пропускает текущую итерацию внутреннего цикла и переходит к следующей. В коде, если **i** равно 1 и **j** равно 1, программа выводит сообщение и продолжает выполнение внешнего цикла.

Таким образом, код демонстрирует контроль над выполнением циклов на основе условий, что позволяет эффективно управлять потоком программы.

# Заключение

• Ключевое слово **if** используется для оценки проверочного выражения на предмет логических значений **true** или **false**.

• Блок операторов **if** может содержать один или более операторов, которые выполняются только тогда, когда проверочное выражение возвращает значение **true**.

• Ключевое слово **else** определяет альтернативные операторы для выполнения программы в случае, когда проверка, выполняемая ключевым словом **if**, возвращает значение **false**.

• Комбинация операторов **if else** позволяет программе осуществлять условное ветвление.

• Оператор **switch** в большинстве случаев предлагает хорошую альтернативу операторам **if else**, предоставляя различные опции для выбора.

• Каждая опция **case** может быть завершена ключевым словом **break**, и, таким образом, выполняться будут только операторы, связанные с этой опцией.

• Ключевое слово **default** может определять операторы, исполняемые в случае, когда все опции **case** возвращают значение **false**. • Цикл с периодичностью исполняет операторы, содержащиеся в нем, до тех пор, пока проверочное выражение не возвратит значение **false**.

• После ключевого слова **for** в скобках указываются инициализатор, проверочное выражение и модификатор счетчика.

• В циклах **while** и **do-while** во избежание входа в бесконечный цикл нужно изменять значение переменной, используемой в проверочном выражении.

• В циклах **for** и **while** проверочное выражение оценивается в самом начале — перед первой итерацией цикла.

• В циклах **do-while** проверочное выражение оценивается в конце цикла — после первой итерации.

• Итерацию цикла можно пропустить с помощью ключевого слова **continue**.

• Используя ключевое слово **break**, можно завершать работу цикла.

• Вложенные внутренние циклы могут использовать метки вместе с ключевыми словами **break** и **continue**, чтобы передавать управление внешнему циклу