Практическая работа №4

Использование операторов ветвления и выбора при решении вычислительных задач

Одним из фундаментальных элементов многих языков программирования являются условные конструкции. Данные конструкции позволяют направить работу программы по одному из путей в зависимости от определенных условий. В языке Java существуют две условные конструкции: if...else и switch...case.

Выражение if…else проверяет истинность некоторого условия и в зависимости от результатов проверки выполняет определенный код:

```
int num1 = 6;
int num2 = 4;
if(num1>num2){
    System.out.println("Первое число больше второго");
}
```

После ключевого слова if ставится условие. Если это условие выполняется, то срабатывает код, который помещен в далее в блоке if после фигурных скобок. В качестве условий выступает операция сравнения двух чисел.

Так как в данном случае первое число больше второго, то выражение num1 > num2 истинно и возвращает логическое значение true. Следовательно, управление переходит в блок кода после фигурных скобок и начинает выполнять содержащиеся там инструкции, а конкретно – метод

```
System.out.println("Первое число больше второго");
```

Если бы первое число оказалось бы меньше второго или равно ему, то инструкции в блоке if не выполнялись бы.

Если мы захотим, чтобы при несоблюдении условия также выполнялись какие-либо действия, мы можем добавить блок else:

```
int num1 = 6;
int num2 = 4;
if(num1>num2){
    System.out.println("Первое число больше второго");
}
else{
    System.out.println("Первое число меньше второго");
}
```

Но сравнение чисел предполагает три результата: первое число больше второго, первое число меньше второго, и числа равны. С помощью выражения else…if мы можем обрабатывать дополнительные условия:

```
int num1 = 6;
int num2 = 8;
if(num1>num2){
    System.out.println("Первое число больше второго");
}
else if(num1<num2){
    System.out.println("Первое число меньше второго");
}
else{
    System.out.println("Числа равны");
}
```

Также мы можем объединить несколько условий, используя логические операторы:

```
int num1 = 8;
int num2 = 6;
if(num1 > num2 && num1 > 7){
    System.out.println("Первое число больше второго и больше 7");
}
```

Блок if будет выполняться, если num1 > num2 равно true и одновременно num1 > 7 равно true.

Конструкция switch…case аналогична конструкции if…else…if, так как позволяет обработать сразу несколько условий:

```
int num = 8;
switch(num){
    case 1:
        System.out.println("число равно 1");
        break;
    case 8:
        System.out.println("число равно 8");
        num++;
        break;
    case 9:
        System.out.println("число равно 9");
        break;
    default:
        System.out.println("число не равно 1, 8, 9");
}
```

После ключевого слова switch в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после операторов case. И если совпадение найдено, то будет выполнен соответствующий блок case.

В конце блока case ставится оператор break, чтобы избежать выполнения других блоков. Например, если бы убрали оператор break в следующем случае:

```
case 8:
System.out.println("число равно 8");
num++;
case 9:
System.out.println("число равно 9");
break;
```

то выполнился бы блок case 8, (поскольку переменная num равна 8). Но, так как в этом блоке оператор break отсутствует, после его выполнения начал бы выполняться блок case 9.

Если мы хотим также обработать ситуацию, когда совпадения не будет найдено, то можно добавить необязательный блок default, как в примере выше.

Также мы можем определить одно действие сразу для нескольких блоков case подряд:

```
int num = 3;
int output = 0;
switch(num){
    case 1:
        output = 3;
        break;
    case 2:
    case 3:
    case 4:
        output = 6;
        break;
    case 5:
        output = 12;
        break;
    default:
        output = 24;
}
System.out.println(output);
```

В Java версии 12 и выше switch…case вместо оператора стал выражением – появилась возможность возвращать результат его работы, который можно присвоить

переменной. Это позволяет существенно сократить и упростить для понимания программный код:

```
int num = 3;
int output = switch (num) {
    case 1 -> 3;
    case 2, 3, 4 -> 6;
    case 5 -> 12;
    default -> 24;
};
System.out.println(output);
```

Практические задания

1. Определить максимальное и минимальное значения из трех различных целых чисел.
2. Известны год и номер месяца рождения человека, а также год и номер месяца сегодняшнего дня (январь – 1, февраль – 2, и т. д.). Определить возраст человека (число полных лет). В случае совпадения номеров месяцев считать, что прошёл целый год.
3. Дано целое число k, представляющее собой номер дня в году ($1 \leq k \leq 365$). Определить, каким будет k-й день года: выходным (суббота и воскресенье) или рабочим, если 1 января этого года – понедельник.
4. В чемпионате по футболу команде за выигрыш дается 3 очка, за ничью – 1 очко, за проигрыш – 0 очков. Известно количество очков, полученных командой за игру. Определить словесный результат игры (выигрыш, проигрыш или ничья).
5. Составить программу, которая в зависимости от порядкового номера месяца (1, 2, ..., 12) выводит на экран:
   а) его название (январь, февраль, ..., декабрь);
   б) время года, к которому относится этот месяц;
   в) количество дней в этом месяце (год считать не високосным).