

Лекция 11

Java. Управляющие конструкции.

Условный оператор if

Формат:

```
if (условное_выражение) { команда1 } else{ команда2 }
```

Выполнение условного оператора начинается с проверки условия. Если условие истинно (равно **true**), выполняются команды, указанные в фигурных скобках сразу после условия. Если условие ложно (равно **false**), то выполняются команды, размещенные в блоке (выделенном фигурными скобками) после ключевого слова **else**. После выполнения условного оператора управление передается следующей после него команде.

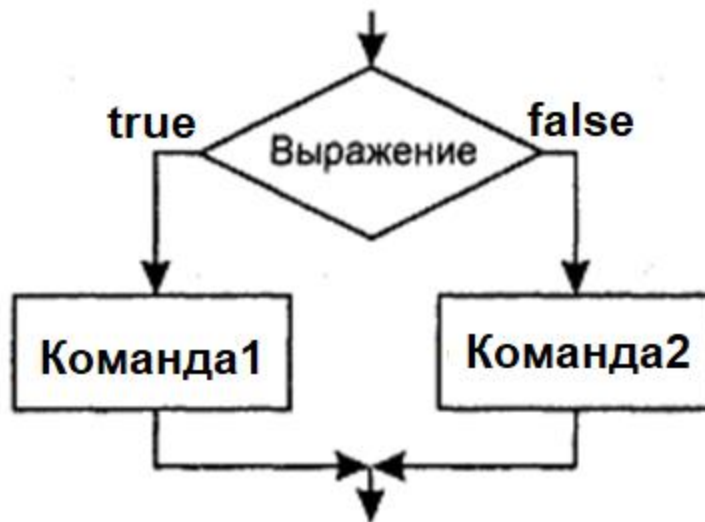
Условный оператор if

Если любой из двух блоков команд (в **if**-блоке или в **else**-блоке) состоит из одной команды, то фигурные скобки для соответствующего блока можно не использовать. Фигурные скобки улучшают читабельность программы.

Сокращенная форма условного оператора:

```
if (условное_выражение) { команда }
```

Структурные схемы:



Условный оператор if - примеры

Пример 1:

```
class demo1{  
    public static void main(String[] args){  
        int n=-100;  
        if (n>0) System.out.println("Положительное");  
        else System.out.println("Отрицательное");  
    }  
}
```

Отрицательное

Пример 2:

```
class demo1{  
    public static void main(String[] args){  
        int a=10, b=35, x;  
        if (a>b) x=a;  
        else x=b;  
        System.out.println("Наибольшее = " + x);  
    }  
}
```

Наибольшее = 35

Пример на вложенные условные операторы

Напишем программу, в которой будем считывать целое число, и в зависимости от значения этого числа выводится то или иное сообщение. Для считывания числа используем консольный ввод.

Для считывания значений, которые пользователь вводит через окно вывода, нам нужен объект класса **Scanner**. Класс импортируется в программу инструкцией **import java.util.Scanner**. Объект класса **Scanner** создается командой:

```
Scanner input=new Scanner(System.in);
```

Объект класса **Scanner** создается с помощью инструкции **new**, после которой указано название класса. Параметром, необходимым для создания объекта, является ссылка **System.in** на объект стандартного потока ввода. Ссылка на созданный объект записывается в объектную переменную **input**.

Командой **System.out.print("Введите целое число: ")** в окно выводится сообщение.

В отличие от метода **println()**, метод **print()** выводит сообщение без перевода на новую строку.

Пример на вложенные условные операторы

```
import java.util.Scanner;
public class demo1{
    public static void main(String[] args){
        int a;
        // Создание объекта класса Scanner:
        Scanner input=new Scanner(System.in);
        System.out.print("Введите целое число: ");
        a=input.nextInt(); // Считывание целого числа
        if(a==0)            // Если введен ноль
            System.out.println("Вы ввели ноль!");
        else if(a==1)       // Если введена единица
            System.out.println("Вы ввели единицу!");
        else if(a%2==0)     // Если введено четное
            System.out.println("Вы ввели четное!");
        else // В прочих случаях
            System.out.println("Вы ввели нечетное!");
        System.out.println("Успешное окончание программы");
    }
}
```

Пример на вложенные условные операторы

Командой `a=input.nextInt()` считывается целочисленное значение, введенное пользователем, которое записывается в переменную `a`.

Из объекта `input` вызывается метод `nextInt()`. Метод в качестве результата возвращает целочисленное значение, которое ввел пользователь (т.е. появляется сообщение «Введите целое число:», и курсор находится в области вывода. Пользователь вводит число, нажимает клавишу, и программа считывает введенное пользователем значение).

Для считанного значения отслеживаются варианты действий:

- пользователь ввел ноль;
- пользователь ввел единицу;
- пользователь ввел четное число (делится без остатка на 2);
- пользователь ввел нечетное число (не делится без остатка на 2).

Пример на вложенные условные операторы

Перебор всех возможных вариантов реализован через блок вложенных условных операторов. Перечисленные ранее условия проверяются по очереди, до первого выполненного условия. Если ни одно из условий не является истинным, то выполняются команды в последнем **else**-блоке вложенных условных операторов. Результаты выполнения программы:

```
Введите целое число: 1
Вы ввели единицу!
Успешное окончание программы
```

```
Введите целое число: 25
Вы ввели нечетное!
Успешное окончание программы
```

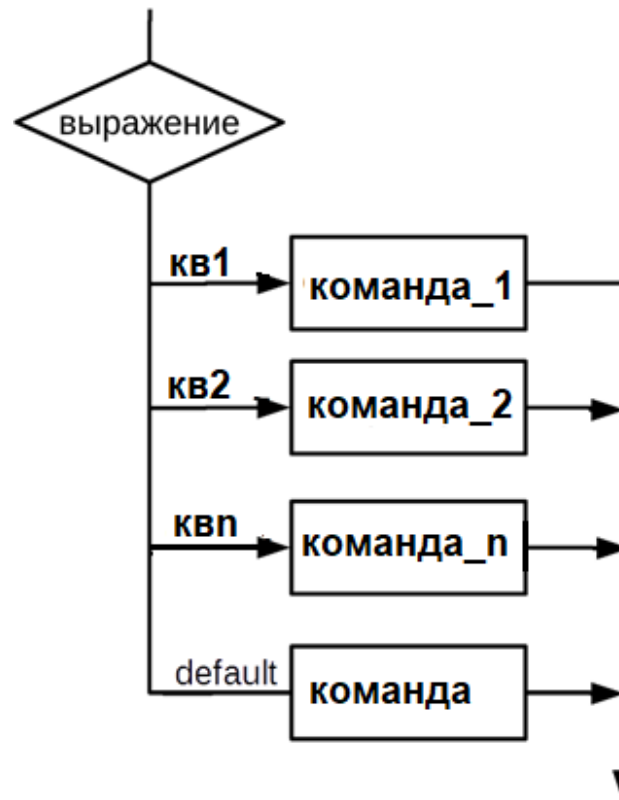
```
Введите целое число: 0
Вы ввели ноль!
Успешное окончание программы
```

```
Введите целое число: 24
Вы ввели четное!
Успешное окончание программы
```


Оператор выбора switch

Оператор **switch** используется для выбора одной из нескольких ветвей алгоритма. Формат:

```
switch (выражение) {  
case константное_выражение_1: [команда_1]; break;  
case константное_выражение_2: [команда_2]; break;  
case константное_выражение_n: [команда_n]; break;  
[default: команда ];  
}
```



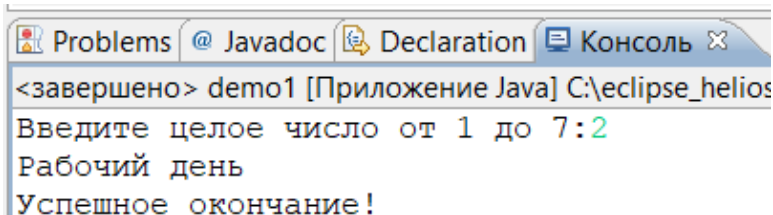
Оператор выбора **switch**

Алгоритм работы оператора выбора: сначала вычисляется выражение в **switch**-инструкции. Затем вычисленное значение последовательно сравнивается, до первого совпадения, с константным выражением, указанным после инструкций **case**. Если совпадение найдено, то начинают выполняться команды соответствующего блока. Выполняются они до первой инструкции **break**, а если таковая не встретится – то до конца оператора выбора. Если при сравнении значения выражения с контрольными значениями совпадения нет, то выполняются команды в блоке **default** (если он есть).

Пример 1: напомним программу чтения целого числа от 1 до 7 и печати «рабочий» или «выходной» день, используя оператор выбора.

Оператор выбора – пример

```
import java.util.Scanner;
public class demo1 {
public static void main(String[] args){
    int x;
    // Создание объекта класса Scanner:
    Scanner input=new Scanner(System.in);
    System.out.print("Введите целое число от 1 до 7:");
    x=input.nextInt(); // Считывание целого числа
    switch(x) {
    case 1: case 2: case 3: case 4: case 5:
        System.out.println("Рабочий день"); break;
    case 6: case 7:
        System.out.println("Выходной день"); break;
    default: System.out.println("Ошибка");
    }
    System.out.println("Успешное окончание!");
}
```



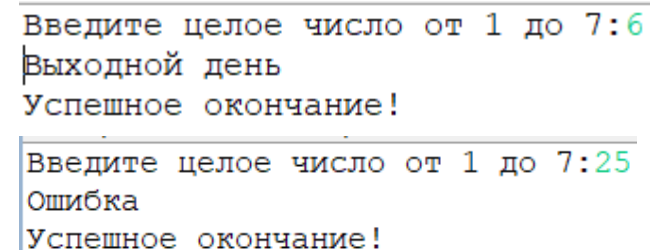
Problems @ Javadoc Declaration Консоль X

<завершено> demo1 [Приложение Java] C:\eclipse_helios

Введите целое число от 1 до 7:2

Рабочий день

Успешное окончание!



Введите целое число от 1 до 7:6

Выходной день

Успешное окончание!

Введите целое число от 1 до 7:25

Ошибка

Успешное окончание!

Оператор выбора – пример

Если пропустить первый break и ввести 2, то получим:

```
Введите целое число от 1 до 7:2  
Рабочий день  
Выходной день  
Успешное окончание!
```

Если пропустить оба break и ввести 2, то получим:

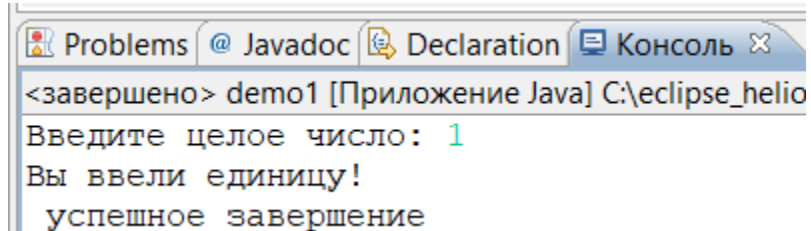
```
Введите целое число от 1 до 7:2  
Рабочий день  
Выходной день  
Ошибка  
Успешное окончание!
```

Получили «эффект проваливания»!!!

Пример 2: напишем программу чтения целого числа и печати какое это число (0 или 1 или четное или нечетное), используя оператор выбора.

Оператор выбора – пример

```
import java.util.Scanner;
class demo1{
    public static void main(String[] args){
        int a; // Создание объекта класса Scanner:
        Scanner input = new Scanner(System.in);
        System.out.print("Введите целое число: ");
        a=input.nextInt(); // Считывание целого числа:
        switch(a){
            case 0: System.out.println("Вы ввели ноль!"); break;
            case 1: System.out.println("Вы ввели единицу!"); break;
            default:
                switch(a%2) { // Если введено четное число
                    case 0: System.out.println("Вы ввели четное число!");
                        break;
                    default: System.out.println("Вы ввели нечетное число!");
                }
        }
        System.out.println(" успешное завершение");
    }
}
```



Problems Javadoc Declaration Консоль X

<завершено> demo1 [Приложение Java] C:\eclipse_helio

Введите целое число: 1
Вы ввели единицу!
успешное завершение

Введите целое число: 25
Вы ввели нечетное число!
успешное завершение

Введите целое число: 22
Вы ввели четное число!
успешное завершение

Циклы

Для выполнения многократно повторяющихся действий используют операторы цикла. В **Java** существует несколько операторов цикла:

- цикл с параметром **for**;
- цикл с предусловием **while**;
- цикл с постусловием **do-while**.

Цикл for

Цикл с параметром имеет следующий формат:

for (выражение1; выражение2; выражение3) команда;

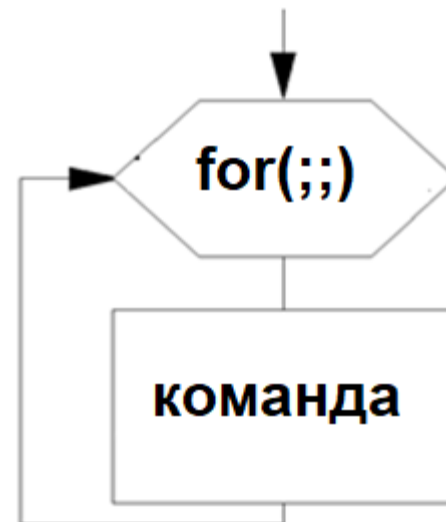
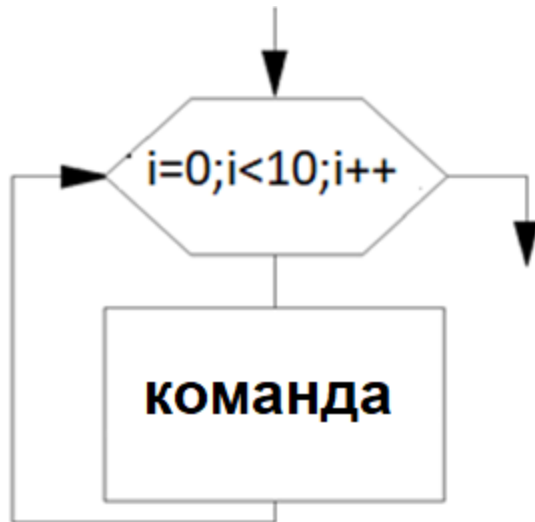
Выражения разделяются точкой с запятой.

Выражение 1 – инициализация;

выражение 2 – условие окончания цикла;

выражение 3 – обновление.

Структурные схемы:



Цикл for

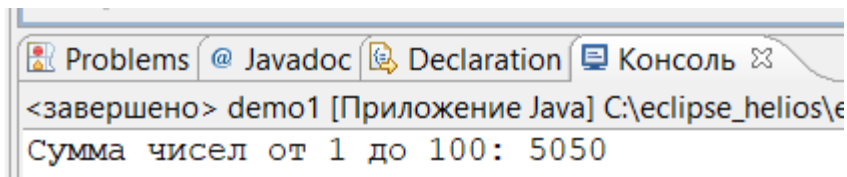
Алгоритм работы:

Сначала выполняется инициализация первого выражения (только один раз в самом начале работы оператора цикла). Затем проверяется условие во второго выражения. Если оно истинно (значение **true**), то выполняются команды в теле оператора цикла (команды в фигурных скобках). Далее выполняются обновление – третье выражение в круглых скобках. Проверяется условие второго выражения. Если условие истинно, то выполняются команды в теле оператора цикла и третье выражение , и затем снова проверяется условие, и так далее. □ Если при проверке условия окажется, что оно ложно (значение **false**), то выполнение оператора цикла на этом завершается.

Рассмотрим на примере:

Цикл for - пример

```
class demo1{  
    public static void main(String[] args){  
  
        int i, n=100; // Индекс и верхняя граница суммы  
        int sum=0; // Переменная для записи значения суммы  
  
        for(i=1;i<=n;i++){ // ЦИКЛ  
            sum+=i;  
        }  
        System.out.println( "Сумма чисел от 1 до "+n+": "+sum );  
    }  
}
```



Цикл for - примеры

Пример 1:

```
for(float i=1.0f;i<=10.2f;i+=2.5f) {  
    System.out.println( "Числа "+i );}
```

```
Числа 1.0  
Числа 3.5  
Числа 6.0  
Числа 8.5
```

Пример 2:

```
for(char i='0';i<='9';i+=2) {  
    System.out.println( "СИМВОЛ "+i );  
}
```

```
СИМВОЛ 0  
СИМВОЛ 2  
СИМВОЛ 4  
СИМВОЛ 6  
СИМВОЛ 8
```

Пример 3:

```
for(char i='и';i>='в';i-=1) {  
    System.out.println( "СИМВОЛ "+i );  
}
```

```
СИМВОЛ и  
СИМВОЛ э  
СИМВОЛ ж  
СИМВОЛ е  
СИМВОЛ д  
СИМВОЛ г  
СИМВОЛ в
```

Пример 4:

```
int sum,i,n;
```

```
Сумма нечетных чисел от 1 до 100: 2500
```

```
for(sum=0,i=1,n=100; i<=n; sum+=i,i+=2);
```

```
System.out.println("Сумма нечетных чисел от 1 до "+n+": "+sum);
```

Цикл for - примеры

Пример 5:

```
public class demo1 {  
    public static void main(String[] args) {  
        int sum=0,i=1,n=100;  
        for(;;) {           // В операторе цикла все блоки пустые  
            sum+=i;  
            i+=2;  
            if(i>n) break; // Завершение оператора цикла  
        }  
        System.out.println("Сумма нечетных чисел от 1 до "+n+":  
"+sum);  
        System.out.println("Успешное окончание!");  
    }  
}
```

```
Сумма нечетных чисел от 1 до 100: 2500  
Успешное окончание!
```

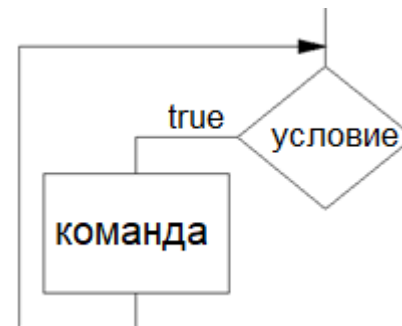
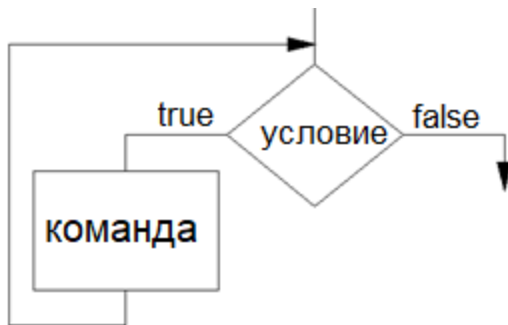
Цикл while

Формат цикла с предусловием

while (выражение) команда;

После ключевого слова **while** в круглых скобках указывается условие. Оно проверяется в начале выполнения оператора цикла **while**. Если условие истинно, то выполняются команды в теле оператора цикла (если их несколько, то они заключаются в фигурные скобки). После этого снова проверяется условие. Если оно истинно, то вновь выполняются команды и проверяется условие, и так далее. Работа оператора цикла завершается, если при очередной проверке условия оно окажется ложным.

Структурная схема:



Цикл while - пример

Пример – поиск нечетных чисел от 1 до 100:

```
public class demo1 {  
    public static void main(String[] args) {  
        int sum=0,i=1,n=100;  
        while(i<=n) { // Оператор цикла  
            sum+=i;  
            i+=2;  
        }  
        System.out.println("Сумма нечетных чисел от 1 до  
"+n+": "+sum);  
    }  
}
```

или можно было так:

```
while(true) { // вечный цикл  
    sum+=i;  
    i+=2;  
    if (i>=n) break;  
}
```

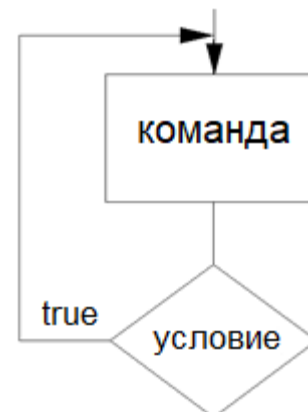
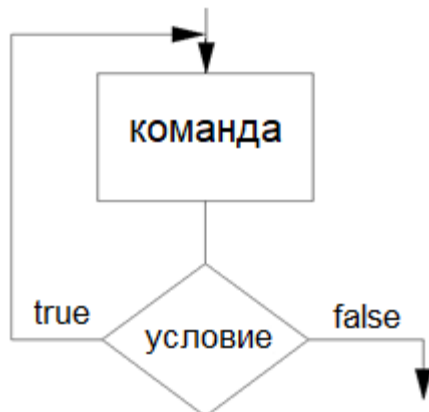
Цикл do-while

Цикл с постусловием и имеет вид:

```
do {команда;  
   } while (выражение);
```

Выполнение оператора начинается с блока команд, размещенных в фигурных скобках после ключевого слова **do**. Затем проверяется условие, указанное в круглых скобках после ключевого слова **while**. Если условие истинно, то снова выполняются команды и затем проверяется условие, и так далее. Тело цикла хотя бы один раз будет выполнено!!!!

Структурная схема:



Цикл do-while - пример

```
public class demo1 {  
    public static void main(String[] args) {  
        int sum=0,i=1,n=100;  
        do{  
            sum+=i;  
            i+=2;  
        }while(i<=n) ;  
        System.out.println("Сумма нечетных чисел от 1 до  
"+n+": "+sum); |Сумма нечетных чисел от 1 до 100: 2500  
    }  
}
```

ИЛИ МОЖНО ТАК:

```
do{  
    sum+=i;  
    i+=2;  
    if (i>=n) break;  
}while(true) ;
```

Пример - решение квадратного уравнения

Рассмотрим программу, в которой решается квадратное уравнение, то есть уравнение вида $ax^2 + bx + c = 0$.

В общем случае корнями уравнения являются значения $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ при условии, что соответствующие значения вычислимы.

Частные случаи:

- параметр $a = 0$; уравнение не является квадратным – отсутствует слагаемое с x^2 – это линейное уравнение вида $bx + c = 0$;
- если параметр b отличен от нуля (при условии, что $a = 0$), то уравнение имеет решение $x = -c/b$. Если же $b = 0$, то возможны два варианта: отсутствие решения при $c \neq 0$ или решение – любое число, если $c = 0$;
- в случае, если параметр $a \neq 0$, выделим три ситуации, определяемые знаком дискриминанта $D = b^2 - 4ac$. При $D < 0$ квадратное уравнение на множестве действительных чисел решений не имеет. Если $D = 0$, квадратное уравнение имеет единственный корень $x = -\frac{b}{2a}$

При $D > 0$ уравнение имеет два корня – это $x = -\frac{b \pm \sqrt{D}}{2a}$.

Пример - решение квадратного уравнения

```
import java.util.Scanner;
public class demo1 {
public static void main(String[] args){
    Scanner input=new Scanner(System.in);
    double a,b,c;           // Параметры уравнения
    double x1,x2,D;         // Корни и дискриминант
    System.out.println("Параметры уравнения:");
    System.out.print("a=");  a=input.nextDouble();
    System.out.print("b=");  b=input.nextDouble();
    System.out.print("c=");  c=input.nextDouble();
    // Поиск решения:
    if(a==0){                // Если a равно 0
        System.out.println("Линейное уравнение!");
        if(b!=0){            // Если a равно 0 и b не равно 0
            System.out.println("Решение  $x = " + (-c/b) + " . "$ ");
        }else{
            if(c==0){         // Если a, b, и c равны нулю
                System.out.println("Решение — любое число.");
            }else{           // Если a и b равны нулю, а c — нет
                System.out.println("Решений нет!");
            }
        }
    }
}
```

Пример - решение квадратного уравнения

```
else{           // Если а не равно 0
    System.out.println("Квадратное уравнение!");
    // Дискриминант (значение):
    D=b*b-4*a*c;
    if(D<0){     // Отрицательный дискриминант
        System.out.println("Действительных решений нет!");
    }else{       // Нулевой дискриминант
        if(D==0){
            System.out.println("Решение x="+(-b/2/a));
        }else{  // Положительный дискриминант
            x1=(-b-Math.sqrt(D))/2/a;
            x2=(-b+Math.sqrt(D))/2/a;
            System.out.println("Два решения: x="+x1+" и x="+x2+".");
        }
    }
}
System.out.println("Работа программы завершена.");
}
```

Пример - решение квадратного уравнения

Результат выполнения программы:

Параметры уравнения:

a=0

b=4

c=-64

Линейное уравнение!

Решение x=16.0.

Работа программы завершена.

Параметры уравнения:

a=-3

b=-12

c=-345

Квадратное уравнение!

Действительных решений нет!

Работа программы завершена.

Параметры уравнения:

a=2

b=4

c=-66

Квадратное уравнение!

Два решения: x=-6.830951894845301 и x=4.830951894845301.

Работа программы завершена.