



Java

Операции.

Управляющие конструкции

Лекция #2

Пустовалова О.Г.
доцент. каф. мат.мод.
ИММИКН ЮФУ

Содержание



Арифметические операции



Логические операции



Операции сравнения



Основные операторы



Примеры

АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

Сокращенная запись бинарных арифметических операций

```
int a = 1;  
int b = 2;  
int c = 20;  
  
a += 5;  
b *= 4;  
c += a * b;  
c %= 6; // остаток от деления
```

// 9/2=4 - целое разделить на целое будет целое

```
System.out.println("a = " + a);  
System.out.println("b = " + b);  
System.out.println("c = " + c);
```

```
a = 6  
b = 8  
c = 2
```

Неявное приведение типов

```
int x=2;
```

```
x+=3.5;
```

```
System.out.println(x);
```

5

```
char c='c';
```

```
c-=2.0;
```

```
System.out.println(c);
```

a

Во избежание ошибок приводить типы всегда лучше явно!

Операции инкремента и декремента

```
double x=5.3;
```

```
// операции инкремента
```

```
System.out.println(x++); →
```

```
System.out.println(++x); →
```

5.3

7.3

```
// операции декремента
```

```
System.out.println(x--); →
```

```
System.out.println(--x); →
```

7.3

5.3

Остаток от деления

// остаток от деления

System.**out**.println(5.3%5);



0.29999999999999998

System.**out**.println(5.3%2);



1.2999999999999998

// деление

System.**out**.println(5.3/5);



1.06

System.**out**.println(**(int)**5.3/5);



1

ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Логические операции

		NOT	AND	OR	XOR
a	b	!a	a&b	a b	a^b
true	true	false	true	true	false
true	false	false	false	true	true
false	true	true	false	true	true
false	false	true	false	false	false

Вывести таблицу логических операций.

Логические операции сокращенного вычисления

&& - сокращенная конъюнкция (conditional-AND)

|| - сокращенная дизъюнкция (conditional-OR)

Правый операнд сокращенных операций вычисляется только в том случае, если от него зависит результат операции, т.е. если левый операнд конъюнкции имеет значение true, или левый операнд дизъюнкции имеет значение false.

true && X  вычисляется

false || X  вычисляется

Придумать и реализовать примеры с использованием сокращенной формы логических выражений.

ОПЕРАЦИИ СРАВНЕНИЯ

Операции сравнения

больше	>
меньше	<
больше или равно	>=
меньше или равно	<=
равно	==
не равно	!=

В отличие от Python такое сравнение **$a < x < b$** напрямую записать нельзя. Нужно писать так:

$(a < x) \&\& (x < b)$

или так:

$a < x \&\& x < b$

(приоритет операций сравнения выше чем приоритет логических операций)

Приоритет операций

1. Постфиксные операции ++ и --.
2. Префиксные операции ++ и --, дополнение ~ и отрицание !.
3. Приведение типа (тип).
4. Умножение *, деление / и взятие остатка %.
5. Сложение + и вычитание -.
6. Сдвиги <<, >>, >>>.
7. Сравнения >, <, >=, <=.
8. Сравнения ==, !=.
9. Побитовая конъюнкция &.
10. Побитовое исключающее ИЛИ ^.
11. Побитовая дизъюнкция |.
12. Конъюнкция &&.
13. Дизъюнкция ||.
14. Условная операция ?::.
15. Присваивания =, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=.

ОСНОВНЫЕ ОПЕРАТОРЫ

Основные операторы

- `if` - условный оператор;
- `while`, `do-while`, `for` - операторы цикла;
- `switch` - оператор выбора;
- `break`, `continue` и `return` - операторы перехода;
- `{}` - блок;
- `;` - пустой оператор.

Область действия блоков

```
public static void main(String[] args)
{
    int n;
    {
        int k;

    } // переменная k определена только в этом блоке
}
```

Переменные с одинаковым именем

```
public static void main(String [] args)
{
    int n;
    {
        int k;
        int n; // ОШИБКА: переопределить переменную n
              // во внутреннем блоке нельзя
    }
}
```

В языке Java нельзя объявлять переменные с одинаковым именем в двух вложенных блоках!

В языке C++ переменные во вложенных блоках можно переопределять. Внутреннее определение маскирует внешнее.

Это может привести к ошибкам, поэтому в Java подобный подход не реализован.

Пример. Условный оператор

```
double x=100, y=21, z;

if(x>y)
    z=x;
else z=y;

System.out.println(z);
```

100.0

```
double x=100, y=21, z;

if(x>y)
    {z=x;}
else {z=y;}

System.out.println(z);
```

```
double x=100, y=21, z;

if x>y
else {z=y;}

System.out.println(z);
```

'(' expected

```
double x=100, y=21, z;

if (x>y)
    x=x*x;
    z=x;
else {z=y;}

System.out.println(z);
```

'else' without 'if'

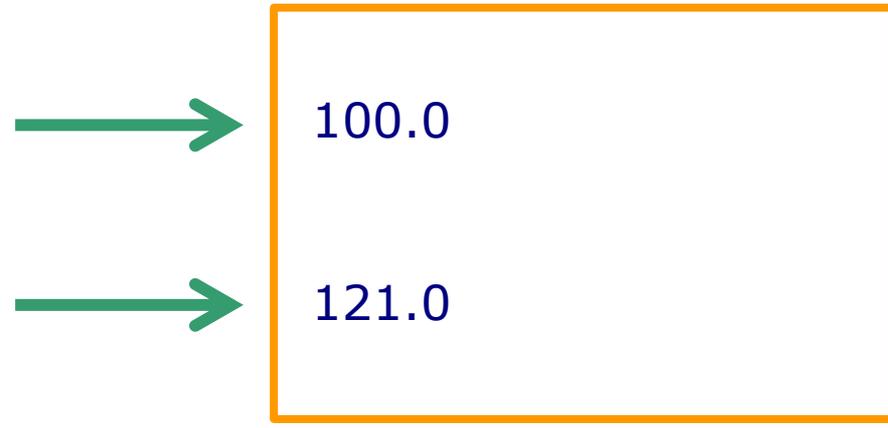
Условие должно быть заключено в скобки!

Пример. Условный оператор. if ... else if

```
if (yourSales >= 2*target)
{
    performance = "Excellent";
    bonus = 1000;
}
else if (yourSales >= 1.5*target)
{
    performance = "Fine";
    bonus = 500;
}
else if (
    yourSales >= target)
{
    performance = "Satisfactory";
    bonus = 100;
}
else
{
    System.out.println("You're fired");
}
```

Тернарный оператор

```
public class Main {  
  
    public static void main(String args[]) {  
        double x=100, y=21;  
  
        // тернарный оператор  
        System.out.println(x > 0 ? x : 0);  
  
        double z=(x > y ? x + y : x - y);  
        System.out.println(z);  
    }  
}
```



100.0

121.0

Оператор множественного выбора **switch**

```
for(int i=0; i<6; i++)  
  switch(i) {  
    case 0:  
      System.out.println("i is zero.");  
      break;  
    case 1:  
      System.out.println("i is one.");  
      break;  
    case 2:  
      System.out.println("i is two.");  
      break;  
    case 3:  
      System.out.println("i is three.");  
      break;  
    default:  
      System.out.println("i is greater than 3.");  
  }
```

i is zero.
i is one.
i is two.
i is three.
i is greater than 3.
i is greater than 3.

Оператор множественного выбора **switch**

```
String str = "two";  
  
switch(str) {  
    case "one":  
        System.out.println("one");  
        break;  
    case "two":  
        System.out.println("two");  
        break;  
    case "three":  
        System.out.println("three");  
        break;  
    default:  
        System.out.println("no match");  
        break;  
}
```



two

Оператор **while**

```
int n = 4;

while(n > 0) {
    System.out.println("tick " + n);
    n--;
}
```

```
tick 4
tick 3
tick 2
tick 1
```

```
int n = 4;

do {
    System.out.println("tick " + n);
    n--;
} while(n > 0);
```

Оператор **for**

```
for (int i = 1; i <= 10; i++)  
    System.out.println(i);
```

Несмотря на то что в Java, как и в C++, отдельными частями оператора цикла **for** могут быть практически любые выражения, существуют неписанные правила, согласно которым все три части оператора цикла **for** должны только инициализировать, проверять и обновлять один и тот же счетчик.

```
for (int i = 10; i > 0; i-- )  
    System.out.println("Counting down . . . " + i) ;  
System.out.println("Blastoff!");
```

Оператор **for**

Будьте внимательны, проверяя в цикле равенство двух чисел с плавающей точкой.

Так, приведенный ниже цикл может вообще не завершиться.

```
for (double x = 0; x != 10; x += 0.1) ...
```

Из-за ошибок округления окончательный результат никогда не будет получен.

В частности, переменная x изменит свое значение с 9.999999999999999998 сразу на 10.099999999999999998 , потому что для числа 0.1 не существует точного двоичного представления.

Оператор **for**

В частности, если переменная определена в операторе цикла `for`, ее нельзя использовать за пределами этого цикла.

Следовательно, если требуется использовать конечное значение счетчика за пределами цикла `for`, соответствующую переменную следует объявить до начала цикла!

```
int i;
```

```
for (i = 1; i <= 10; i++)
```

```
{
```

```
}
```

```
// здесь переменная i по-прежнему доступна
```

Оператор for

```
int a, b;  
  
b = 4;  
for(a=1; a<b; a++) {  
    System.out.println("a = " + a);  
    System.out.println("b = " + b);  
    b--;  
}
```

```
a = 1  
b = 4  
a = 2  
b = 3
```

```
int a, b;  
for(a=1, b=4; a<b; a++, b--) {  
    System.out.println("a = " + a);  
    System.out.println("b = " + b);  
}
```

Использование запятых

Оператор **for**

```
int i;  
boolean done = false;  
  
i = 0;  
for( ; !done; ) {  
  
    System.out.println("i is " + i);  
    if(i == 3) done = true;  
    i++;  
  
}
```

```
i is 0  
i is 1  
i is 2  
i is 3
```

Аналог оператора `foreach`

```
int nums[] = { 1, 2, 3, 4, 5 };
```

```
int sum = 0;
```

```
// use for-each style for to display and sum the values
```

```
for(int x : nums) {
```

```
    System.out.println("Value is: " + x);
```

```
    sum += x;
```

```
}
```

```
System.out.println("Summation: " + sum);
```

```
Value is: 1
```

```
Value is: 2
```

```
Value is: 3
```

```
Value is: 4
```

```
Value is: 5
```

```
Summation: 15
```

Аналог оператора `foreach`

```
int nums[] = { 1, 2, 3, 4, 5 };
```

```
for(int x : nums) {  
    System.out.print(x + " ");  
    x = x * 10; // no effect on nums  
}
```

```
System.out.println();
```

```
for(int x : nums)  
    System.out.print(x + " ");
```



1 2 3 4 5



1 2 3 4 5

Оператор `continue`

```
for(int i=0; i<10; i++) {  
    if (i%2 == 0) continue;  
    System.out.print(i + " ");  
}
```



1 3 5 7 9

Оператор **break**

```
for(int i=0; i<10; i++) {  
    int k=(int) (Math.random() * 10); ;  
    if (k == 5) break;  
    System.out.print(k + " ");  
}
```



7 0 0 4

Оператор **break** с меткой

в Java поддерживается оператор `break` с меткой, обеспечивающий выход из вложенных циклов.

С его помощью можно организовать прерывание глубоко вложенных циклов при нарушении логики управления программой.

метка:

```
{  
    if (условие) break метка; // выход из блока  
}
```

// При выполнении оператора `break` управление передается в эту точку

Оператор **return**

```
boolean t = true;
```

```
System.out.println("Before the return.");
```

Before the return.

```
if(t) return; // return to caller
```

```
System.out.println("This won't execute.");
```



Спасибо за внимание!