# Преобразование типов: явное и неявное

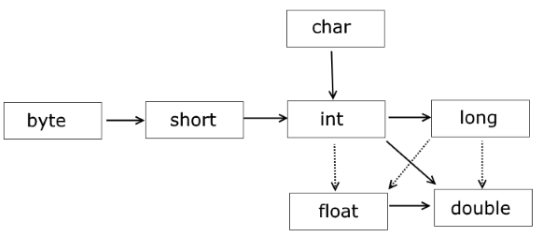
В Java существуют разные типы данных: int, double, char, string и т.д. Чтобы переменную перевести из одного типа в другой, необходимо произвести преобразование типов данных.

Всего существует два типа преобразования: Автоматическое (неявное) и приведение типов (явное).

**Автоматически** происходят расширяющие преобразования. Например:  
byte b = 8;

int d = b; // преобразование от byte к int

Расширяющие преобразования происходят по следующей схеме:



**Приведение типов** или явное преобразование. Обычно это сужающие преобразования типов.

Например:

int a = 5;   
byte b = (byte) a;

# Консольный ввод посредством Scanner

В Java консольный ввод реализуется с помощью Scanner.

Чтобы его использовать, сначала нужно его импортировать с помощью строки “import java.util.Scanner;” (она должна находиться между названием пакета и объявлением класса в программе)

Далее, в методе main нужно создать экземпляр класса scanner. Scanner scanner = new Scanner(System.in);

Теперь можно создавать переменные, значения которым можно присвоить из консоли. Например:  
int x = scanner.nextInt(); /\* вводим целое число x через scanner\*/

double y = scanner.nextDouble(); /\* вводим вещественное число y (в консоли вводится через запятую, а не точку) \*/

Для вывода информации в консоль используются методы System.out.println и если нужно вывести разнородные данные, используется метод System.out.printf();

# Консольный ввод посредством BufferedReader

Для консольного ввода также можно использовать BufferedReader, который имеет гораздо лучшую производительность, чем Scanner. Использование BufferedReader требует импорта пакета java.io в заголовке файла (“import java.io.\*”).

Создают переменную:

BufferedReader br = new BufferedReader (new InputStreamReader(System.in));

InputStreamReader переводит байты в символы, но позволяет считывать при этом только один символ; а BufferedReader позволяет считывать несколько символов. Получается, что символы заносятся в буфер, откуда и считываются.

Затем можно вывести полученную строку: “System.out.println(br.readLine());”

# Форматированный вывод

Для форматирования в Java существует класс Formatter, включенный в пакет java.util. У него есть метод format(), который преобразует переданные в него параметры в строку заданного формата.

Например:

import java.util.Formatter;

public class FormatStr {

public static void main(String[] args) {

String s = “ %t получена сумма в размере %f по счёту N %s ”;

String answer = String.format(s, new Date(), 72.5, “AB137”)

//%t – соответствует времени

//%f – число float

//%s - строка

}

}

Также метод format есть у объекта String. Пример:

String str = String.format("Привет - %s! Как дела %s?", "Саша", "на работе"); System.out.println(str);

# Операции в Java

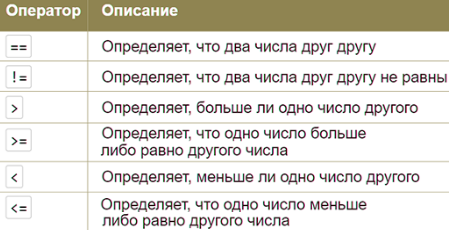
В Java есть много операций. Например, операция присваивания обозначается символом “=”.

x=3; a=x;

Также есть и арифметические операции: сложение чисел (+), вычитание (-), умножение (\*), деление (/), остаток от деления (%).

Есть операторы инкремента (++) и декремента (--). Они могут быть как префиксными, так и постфиксными.

Также есть операторы сравнения:



Логические операторы:

! — оператор отрицания;

&& — оператор логическое И (сокращенный);

|| — оператор логическое ИЛИ (сокращенный);

& — оператор побитовое И;

| — оператор побитовое ИЛИ;

^ — оператор побитовое исключающее ИЛИ.

В Java также используются поразрядные операторы:

~ поразрядный оператор отрицания

>> побитовый сдвиг вправо

>>> беззнаковый побитовый сдвиг вправо

<< побитовый сдвиг влево

Их используют для кодирования/декодирования, шифрования, генерации случайных чисел.

# Оператор If

Оператор If используются для создания условия и ветвления программы. Можно создать конструкцию вида:

if (условие){

оператор 1 // если условие истинно, то выполняется оператор 1

}

else

{

оператор 2 // если условие не верно, то выполняется оператор 2

}

# Операторы break и continue

Оператор break чаще всего используется для выхода из цикла. Когда оператор break встречается внутри цикла, второй заканчивается и программное управление передается оператору, следующему за ним

// Использование break для выхода из цикла.

public class BreakLoop {

public static void main(String[] args) {

for (int i = 0; i < 100; i++) {

if (i == 5) {

break; // завершить цикл, если i = 5 }

System.out.println("i: " + i); }

System.out.println("Цикл завершен."); }

}

Оператор continue используется чтобы «отбросить» следующий за ним код и сразу перейти к следующей итерации цикла.

public class Continue {

public static void main(String[] args) {

for (int i = 0; i < 10; i++) {

System.out.print(i + " ");

if (i % 2 == 0) {

continue; //если число чётное, то цикл продолжается без печати числа

}

System.out.println("");

}

}

}

# Оператор switch: структура, назначение и примеры

Switch используется для того, чтобы обработать большое количество условий. Вместо If-else можно использовать switch, чтобы придать структурированности и читабельности коду.

Switch имеет следующий вид:

switch (выражение) {

case значение1: // Блок кода 1

break;

case значение2: // Блок кода 2

break;

case значениеN: // Блок кода N

break;

default : // Блок кода для default

}

Значений может быть любое количество. Если не выполняется ни одно из них, то выполняется default.

Пример:

int x = 1;

switch (x) {

case 1:

System.out.println("Один");

break;

case 2:

System.out.println("Два");

break;

default:

System.out.println("Неправильно введено число");

break;

}

Для нескольких значений case можно указать один блок:

int x = 3;

switch (x) {

case 1:

case 2:

case 3:

System.out.println("Введено верное число");

default:

System.out.println("Неправильно введено число");

break;

}