**Очень простая лабораторная работа на повторение.**

# Простые типы данных

В Java существует четыре группы базовых типов: для работы с целыми числами, для работы с числами в формате с плавающей точкой (действительные числа), символы и логический тип — таким образом, всего получается восемь базовых типов. Базовые типы Java перечислены в табл. 1.1.

Таблица 1.1. Базовые (простые) типы в Java

|  |  |  |  |
| --- | --- | --- | --- |
| Тип данных (название) | Количество битов | Пояснение | Класс-оболочка |
| byte | 8 | Целые числа в диапазоне от  -128 до 127 | Byte |
| short | 16 | Целые числа в диапазоне от  -32768 до 32767 | Short |
| int | 32 | Целые числа в диапазоне от  -2147483648 до 2147483647 | Integer |
| long | 64 | Целые числа в диапазоне от  -9223372036854775808 до 9223372036854775807 | Long |
| float | 32 | Действительные числа. По абсолютной величине изменяются в диапазоне от 3,4 × 10-38 до 3,4 × 1038 | Float |
| double | 64 | Действительные числа двойной точности. По абсолютной величине изменяются в диапазоне от 1,7 × 10–308 до 1,7 × 10308 | Double |
| char | 16 | Символьный тип для представления символьных значений (букв). Диапазон значений от 0 до 65536 (каждое значение соответствует определенному символу) | Character |
| boolean | 8[[1]](#footnote-1) | Логический тип данных. Переменная этого типа может принимать два значения: true (истина) и false (ложь) | Boolean |

В этой же таблице приведены названия классов-оболочек для базовых типов. Классы-оболочки используются в тех случаях, когда переменную соответствующего типа необходимо рассматривать как объект.

## Приведение типов данных

Язык программирования Java – язык строгой типизации переменных. При вычислении выражения, в которое входят переменные разных типов, автоматически выполняется преобразование входящих в выражение переменных к общему формату. Процесс автоматического преобразования типов подчиняется нескольким базовым правилам:

* Типы переменных, входящих в выражение, должны быть совместимыми.
* Целевой тип (тип, к которому выполняется приведение) должен быть «шире» исходного типа. Другими словами, преобразование должно выполняться без потери данных
* Перед выполнением арифметической операции типы byte, short и char расширяются до типа int. Если в выражении есть операнды типа long, то расширение осуществляется до типа long. Если в выражении есть операнды типа float, то расширение осуществляется до типа float. Если в выражении есть операнды типа double, то расширение осуществляется до типа double.

В Java предусмотрено явное приведение типов и явное определение типа литерала с помощью **суффиксов типа**. Для приведения выражения к нужному типу перед этим выражением указывается имя типа, заключенное в круглые скобки.

## Ввод и вывод данных

Рассмотрим организацию консольного ввода с помощью класса Scanner.

Для работы с этим классом необходимо включить в заголовок файла программы инструкцию import java.util.\*, то есть подключить (импортировать) пакет java.util.

Общая схема реализации процесса введения данных с консоли посредством класса Scanner такова: на основе стандартного потока ввода System.in создается объект класса Scanner, через который и осуществляется консольный ввод.

Scanner inp =**new** Scanner(System.***in***);

Следующие методы класса Scanner обеспечивают ввод данных:

nextLine() – считывание текстовой строки;

nextInt() – считывание целого числа;

nextDouble() – считывание действительного числа.

Пример:

**import** java.util.Scanner;  
  
**public class** MyClass {  
 **public static void** main(String[] args) {  
 System.***out***.println(**"Input int"**);  
 Scanner inp =**new** Scanner(System.***in***);  
 **int** x=inp.nextInt();  
 System.***out***.println(x);  
 }  
}

## Класс Math

Таблица 1.2. Некоторые методы класса Math

|  |  |
| --- | --- |
| **Оператор класса Math** | **Примечание** |
| **Тригонометрические и обратные тригонометрические функции** | |
| sin(x) | sin(x) – синус |
| cos(x) | cos(x) – косинус |
| tan(x) | tg(x) – тангенс |
| asin(x) | arcsin(x) – арксинус |
| *acos*(x) | arccos(x) – арккосинус |
| atan(x) | arctg(x) – арктангенс |
| atan2(y, x) | Возвращает угол, соответствующий точке с координатами x,y, лежащий в пределах (-\pi,\pi]. |
| toRadians(angdeg) | angdeg / 180.0 \* PI; - перевод углов из градусов в радианы. |
| toDegrees(angrad) | angrad \* 180.0 / PI; - перевод углов из радиан в градусы. |
| **Степени,*****экспоненты*, логарифмы** | |
| exp(x) | e^x - *экспонента* |
| expm1(x) | e^x-1. При x, близком к 0, дает гораздо более точные значения, чем exp(x)-1 |
| log(x) | ln(x) – *натуральный логарифм*. |
| log10(x) | \log_{10}(x) – десятичный логарифм. |
| log1p(x) | \ln(1+x). При x, близком к 0, дает гораздо более точные значения, чем \log(1+x) |
| sqrt(x) | \sqrt{} - квадратный корень |
| cbrt(x) | \sqrt[3]{} - кубический корень |
| hypot(x,y) | \sqrt{x^2+y^2} - вычисление длины гипотенузы по двум катетам |
| pow(x, y) | x^y – возведение x в степень y |
| sinh(x) | \sh(x)=\frac{e^x-e^{-x}}{2} – гиперболический синус |
| cosh(x) | \ch(x)=\frac{e^x+e^{-x}}{2} – гиперболический косинус |
| tanh(x) | \th(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}} – гиперболический тангенс |
| **Модуль, знак, минимальное, максимальное число** | |
| abs(m)  abs(x) | Абсолютное значение числа. Аргумент типа int, long, float или double. Результат того же типа, что аргумент. |
| signum(a)  signum(x) | Знак числа. Аргумент типа float или double. Результат того же типа, что аргумент. |
| min(m,n)  min(x,y) | Минимальное из двух чисел. Аргументы одного типа. Возможны типы: int, long, float, double. Результат того же типа, что аргумент. |
| max(m,n)  max(x,y) | Максимальное из двух чисел. Аргументы одного типа. Возможны типы: int, long, float, double. Результат того же типа, что аргумент. |
| **Округления** | |
| *ceil*(x) | Ближайшее к **x** целое, большее или равное **x** |
| *floor*(x) | Ближайшее к **x** целое, меньшее или равное **x** |
| round(a)  round(x) | Ближайшее к **x** целое. Аргумент типа float или double. Результат типа long, если аргументdouble, и типа int – если float. То же, что (int)*floor*(x + 0.5). |
| rint(x) | Ближайшее к **x** целое. |
| *ulp*(a)  *ulp*(x) | Расстояние до ближайшего большего чем аргумент значения того же типа ("дискретность" изменения чисел в формате с плавающей точкой вблизи данного значения). Аргумент типа float или double. Результат того же типа, что аргумент. |
| **Случайное число, остаток** | |
| random() | *Псевдослучайное число* в диапазоне от 0.0 до 1.0. |
| IEEEremainder(x,y) | Остаток от целочисленного деления **x/y**, то есть **x-y\*n**, где **n** – результат целочисленного деления |

В приведенной таблице величины имеют следующие типы:

x, y, angdeg, angrad – тип double,

a– тип float,

m, n– целые типов long или int.

Математические функции возвращают значения типа double, если в примечании не указано иное.

## Основные операторы для работы с целочисленными величинами

В приведенной ниже таблице (табл. 1.3) i и j обозначают целочисленные выражения, а v – целочисленную переменную.

Таблица 1.3. Базовые целочисленные операции в Java

|  |  |  |  |
| --- | --- | --- | --- |
| **Оператор** | **Название** | **Пример** | **Примечание** |
| **+** | Оператор сложения | i+j | В случае, когда операнды i и j имеют разные типы или типы byte, shortили char, действуют правила *автоматического преобразования* типов. |
| **–** | Оператор вычитания | i-j |
| **\*** | Оператор умножения | i\*j |
| **/** | Оператор деления | i/j | Результат округляется до целого путем отбрасывания дробной части как для положительных, так и для отрицательных чисел. |
| **%** | Оператор остатка от целочисленного деления | i%j | Возвращается остаток от целочисленного деления |
| **=** | Оператор присваивания | v=i | Сначала вычисляется выражение i, после чего полученный результат копируется в ячейку v |
| **++** | Оператор *инкремента* (увеличения на 1) | v++ | v++ эквивалентно v=v+1 |
| **--** | Оператор *декремента* (уменьшения на 1) | v-- | v-- эквивалентно v=v-1 |
| **+=** |  | v+=i | v+=i эквивалентно v=v+i |
| **-=** |  | v-=i | v-=i эквивалентно v=v-i |
| **\*=** |  | v\*=i | v\*=i эквивалентно v=v\*i |
| **/=** |  | v/=i | v/=i эквивалентно v=v/i |
| **%=** |  | v%=i | v%=i эквивалентно v=v%i |

Отдельно выделим важные методы классов Integer и Long:

Функции преобразования строкового значения в числовой формат:

Integer.parseInt( <строка> )

Long.parseLong( <строка> )

Функции Integer.signum (<число>) и Long.signum (<число>) возвращают знак числа, то есть

1, если число положительное,

0, если оно равно 0

-1, если число отрицательное.

**Условный оператор**

Неполный условный оператор:

if(<условие>)

<оператор1>;

Полный условный оператор:

if(<условие>)

<оператор1>;

else

<оператор2>;

**Тернарный оператор if-then-else**

Общая форма оператора if-then-else такова:

<условие> ? <значение1> : <значение2>;

В качестве первого операнда – <условие> – может быть использовано любое выражение, результатом которого является значение типа boolean. Если результат равен true, то тернарный условный оператор возвращает <значение1>, иначе <значение2>. При этом <значение1> и <значение2>, должны возвращать значения одного типа и не должны иметь тип void.

**Оператор выбора switch**

Синтаксис оператора:

switch(<выражение>){

case <значение1>: <последовательность операторов 1;> break;

……………………………

case <значениеN>:<последовательность операторов N;> break;

default:< последовательность операторов;> break;

}

Отметим, что нельзя указывать *диапазон* значений, нельзя перечислять через запятую значения, которым соответствуют одинаковые *операторы*.

**Оператор цикла for**

for(<блок инициализации>; <условие выполнения тела цикла>; <блок изменения счетчиков>)<оператор>;

**Оператор цикла while()**

Синтаксис:

while(<условие>){

// <тело цикла>

}

**Оператор цикла do-while()**

Синтаксис:

do{

// <тело цикла>

}

while(<условие>);

# Объявление и инициализация массивов

Синтаксис объявления массивов в Java совпадает с синтаксисом объявления массивов в языке С# и имеет вид:

//объявление одномерного массива

типЭлементов[] *имя*=**new** типЭлементов[*размер*];

типЭлементов *имя*[]=**new** типЭлементов[*размер*];

При этом допускается указывать квадратные скобки либо после имени типа массива, либо после имени массива. Например, вместо команды int[] nums можно использовать команду int nums[].

В Java многомерные массивы представляют собой массивы массивов. При объявлении переменной многомерного массива для указания каждого дополнительного индекса используют отдельный набор квадратных скобок. //объявление двумерного массива

типЭлементов[][] *имя*=**new** типЭлементов[*размер\_1*][*размер\_2*];

Обращение к элементу одномерного массива осуществляется через имя массива с указанием в квадратных скобках индекса элемента. Индексация элементов массива начинается с нуля.

При объявлении переменной массива, ее можно инициализировать списком значений, указывав (через оператор присваивания) список значений в фигурных скобках. Например: int[] data={3,8,1,7};

## Перебор значений массива

1 способ:

тип[] имя массива = new <тип>[];

for (int i = 0; i < имя.length; i++) {

тело цикла

}

2 способ:

for (тип element : имя массива) {

тело цикла

}

Отметим, что во втором случае мы не имеем доступа к индексу массива.

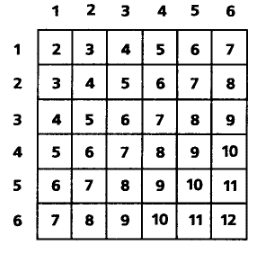
## Класс java.util.Arrays

Класс Arrays предназначен для работы с массивами. Рассмотрим некоторые методы класса Arrays

|  |  |
| --- | --- |
| **Название метода** | **Описание** |
| copyOf | предназначен для копирования массива; |
| copyOfRange | копирует часть массива; |
| toString | позволяет получить все элементы в виде одной строки; |
| sort | сортирует массив методом quick sort; |
| binarySearch | ищет элемент методом бинарного поиска; |
| fill | заполняет массив некоторым значением |
| equals | проверяет массивы на идентичность. Массивы считаются равными, и возвращается true, если они имеют одинаковую длину и равны элементы массивов с одинаковыми индексами |
| deepEquals | проверяет на идентичность массивы массивов; |
| asList | возвращает массив как коллекцию |

**Задачи:**

1. Пользователь интернет-магазина совершает покупку товара, например, ноутбука. Он заказывает определенное количество товара. Написать программу реализующую вывод сообщения о покупке, в зависимости от количества купленного товара. Например, вы купили 1 ноутбук или 3 ноутбука или 15 ноутбуков. Создайте метод, который получает число и возвращает строку.
2. Игра «Длинное число». Двое игроков записывают по очереди n цифр (0..9). Если полученное n-значное число делится на 9, то побеждает игрок, сделавший последний ход, иначе он проигрывает. Написать алгоритм игры пользователя с компьютером.
3. Решить задачу 2 при делении на 7. Использовать класс BigInteger.
4. Написать программу, моделирующую бросание двух костей. Кости – это шестигранные кубики, каждая грань которых имеет значение очков от 1 до 6 включительно. Соответственно, когда мы бросаем две кости, то количество выпавших очков может быть в интервале от 2 до 12 включительно. По логике вещей и теории вероятностей, количество очков, равное 7, должно выпадать чаще других. Т.к. у обеих костей по шесть значений, то всего может быть 36 различных комбинаций бросков. Их можно рассмотреть на рисунке ниже



Примерно в 1/6 всех случаев, у нас должно выпадать 7 очков.

Нужно написать программу, моделирующую бросание двух костей. Для моделирования бросания обоих костей, программа должна использовать генератор случайных чисел rand(). Ваша программа должна бросать две кости ровно 36000 раз. Используйте одномерный массив для подсчета количества раз выпадения той или иной суммы. Определите приемлемость полученных результатов, ориентировочно сравнив их с теоретической вероятностью выпадения того или иного значения. Определите, какое количество очков выпало чаще других.

1. Составьте программу, упорядочивающую по возрастанию/убыванию элементы каждой строки матрицы размером m×n (не писать алгоритм сортировки).

**Дополнительная информация**

**Компиляция и запуск программ из командной строки**

1. В командной строке посмотрите версию Java на вашем компьютере

java –version или javac -version

1. Создайте текстовый файл с именем *MyClass.java* и внесите туда следующий текст:

public class MyClass{

public static void main(String [] args){

System.out.println("!!!");

}

}

Обратите внимание: имя класса и имя файла должны совпадать.

1. Компиляция java-файла. В командной строке перейдите в папку с данным файлом и вызовите команду компиляции:

javac MyClass.java

После успешной компиляции появится файл *MyClass.class*. Расширение ***class*** – это стандартное расширение для файлов с java байт-кодом.

Компиляция со сторонними классами

javac -cp lib.jar MyClass.java

1. Запуск файла *MyClass.class* в виртуальной машине java

java MyClass

Запускается главный класс программы, содержащий метод ***main***.

1. Создание jar-файла (Java Archive – файла, содержащего все классы программы в виде архива)

jar cfe My.jar MyClass MyClass.class

команда имя архива имя главного класса файлы проекта

Запуск архивного файла

java -jar My.jar

или

java -cp My.jar MyClass

где MyClass – это главный класс программы, ключ cp – classpath.

Виртуальной машине нужно указывать дополнительные классы, для этого все используемые jar-файлы перечисляются через точку с запятой

java -cp lib.jar;My.jar MyClass

Просмотр архива, не распаковывая

jar -tf My.jar

Распаковка архива

jar -xf My.jar

1. Для хранения значения этого типа достаточно 1 бита, но в реальности память такими порциями не выделяется, поэтому переменные этого типа могут быть по-разному упакованы виртуальной машиной [↑](#footnote-ref-1)