**Лабораторная работа №1.**

**Основы программирования на языке Java.**

Первая программа на языке Java:

class HelloWorld{

public static void main(String[] args){

System.out.println("Hello, XXI Century World!");

}

}

На этом простом примере можно заметить целый ряд существенных особенностей языка Java.

* Всякая программа представляет собой один или несколько классов, в этом простейшем примере только один *класс* (class).
* Начало класса отмечается служебным словом class, за которым следует имя класса, выбираемое произвольно, в данном случае HelloWorld. Все, что содержится в классе, записывается в фигурных скобках и составляет *тело класса* (class body).
* Все действия производятся с помощью методов обработки информации, коротко говорят просто *метод* (method). Это название употребляется в языке Java вместо названия «функция», применяемого в других языках.
* Методы различаются по именам. Один из методов обязательно должен называться main, с него начинается выполнение программы. В нашей простейшей программе только один метод, значит, имя ему main.
* Как и положено функции, метод всегда выдает в результате (чаще говорят, *возвращает* (return)) только одно значение, тип которого обязательно указывается перед именем метода. Метод может и не возвращать никакого значения, играя роль процедуры, как в нашем случае. Тогда вместо типа возвращаемого значения записывается слово void, как это и сделано в примере.
* После имени метода в скобках, через запятую, перечисляются *аргументы* (arguments) или *параметры* метода.Для каждого аргумента указывается его тип и,через пробел,имя.Впримере только один аргумент, его тип – массив, состоящий из строк символов. Строка символов – это встроенный в Java тип String, а квадратные скобки – признак массива. Имя массива может быть произвольным, в примере выбрано имя args.
* Перед типом возвращаемого методом значения могут быть записаны *модификаторы* (modifiers). В примере их два: слово public означает, что этот метод доступен отовсюду; слово static обеспечивает возможность вызова метода main() в самом начале выполнения программы без создания экземпляра какого-либо класса. Модификаторы вообще необязательны, но для метода main() они необходимы.
* Все, что содержит метод, *тело метода* (method body), записывается в фигурных скобках.

Единственное действие, которое выполняет метод main() в примере, заключается в вызове другого метода со сложным именем System.out.println и передаче ему на обработку одного аргумента, текстовой константы "Hello, 2lth century world!". Текстовые константы записываются в кавычках, которые являются только ограничителями и не входят в состав текста.

Все основные алгоритмические конструкции языка Java (объявление/инициализация/ присваивание переменных, ветвление (операторы if, switch, тернарный оператор), циклы) заимствованы из языка С/С++ и полностью соответствуют установленному там синтаксису.

Простые (или примитивные) типы данных похожи на применяемые в языках С и С++ (см.

табл.1.1 и 1.2).

**Таблица 1.1.** – Целые типы данных.

|  |  |  |
| --- | --- | --- |
| **Тип** | **Разрядность** | **Диапазон** |
|  | **(байт)** |  |
|  |  |  |
| byte | 1 | от -128 до 127 |
|  |  |  |
| short | 2 | от -32768 до 32767 |
|  |  |  |
| int | 4 | от -2147483648 до 2147483647 (−231..231−1) |
|  |  |  |
| long | 8 | от -9223372036854775808 до 9223372036854775807 (−263..263−1) |
|  |  |  |
| char | 2 | от '\u0000' до '\uFFFF' , в десятичной форме от 0 до 65535 |
|  |  |  |

**Таблица 1.2.** – Вещественные типы данных

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип** | **Разрядность** | **Диапазон** | **Точность** |
|  |  |  |  |
| float | 4 | −3.4·1038..3.4·1038 | 7-8 цифр |
|  |  |  |  |
| double | 8 | −1.8·10308..1.8·10308 | 17 цифр |
|  |  |  |  |

**Примечание**:В языкеJavaоперации%(вычисление остатка от деления), ++ (инкремент)и–(декремент) применяются и к вещественным типам.

**Массивы**

Массивы в языке Java относятся к ссылочным типам и описываются своеобразно, но характерно для ссылочных типов. Описание производится в три этапа.

Первый этап – *объявление* (declaration). На этом этапе определяется только переменная типа *ссылка* (reference) *на массив,* содержащая тип массива.Для этого записывается имя типа элементовмассива, квадратными скобками указывается, что объявляется ссылка на массив, а не простая переменная, и перечисляются имена переменных типа ссылка, например,

double[] а, b;

Здесь определены две переменные – ссылки а и b на массивы типа double. Можно поставить квадратные скобки и непосредственно после имени. Это удобно делать среди определений обычных переменных:

int I = 0, ar[], k = -1;

Здесь определены две переменные целого типа i и k, и объявлена ссылка на целочисленный массив ar.

Второй этап – *определение* (installation). На этом этапе указывается количество элементов массива, называемое его *длиной,* выделяется место для массива в оперативной памяти, переменная-ссылка получает адрес массива. Все эти действия производятся еще одной операцией языка Java – операцией new*,* выделяющей участок в оперативной памяти для объекта указанного в операции типа и возвращающей в качестве результата адрес этого участка. Например,

a = new double[5];

b = new double[100];

ar = new int[50];

Индексы массивов всегда начинаются с 0. Массив а состоит из пяти переменных а[0], а[1],…, а[4]. Элемента а[5] в массиве нет. Индексы можно задавать любыми целочисленными выражениями, кроме типа long, например, a[i+j], a[i%5], a[++i]. Исполняющая система Java следит за тем, чтобы значения этих выражений не выходили за границы длины массива.

Третий этап – *инициализация* (initialization). На этом этапе элементы массива получают начальные значения. Например,

а[0] = 0.01; а[1] = -3.4; а[2] = 2.89; а[3] = 4.5; а[4] = -6.7;

for (int i = 0; i < 100; i++) b[i] = 1.0/i;

for (int i = 0; i < 50; i++) ar[i] = 2 \* i + 1;

Первые два этапа можно совместить:

double[] a = new double[5], b = new double[100];

int i = 0, ar[] = new int[50], k = -1;

Можно сразу задать и начальные значения, записав их в фигурных скобках через запятую в виде констант или константных выражений. При этом даже необязательно указывать количество элементов массива, оно будет равно количеству начальных значений:

double[] а = {0.01, -3.4, 2.89, 4.5, -6.7};

Можно совместить второй и третий этап:

a = new double[] {0.1, 0.2, -0.3, 0.45, -0.02};

Можно даже создать безымянный массив, сразу же используя результат операции new, например, так:

System.out.println(new char[] {'H', 'e', '1', '1', 'o'});

Ссылка на массив не является частью описанного массива, ее можно перебросить на другой массив того же типа операцией присваивания. Например, после присваивания а = b обе ссылки а и b указывают на один и тот же массив из 100 вещественных переменных типа double и содержат один и тот же адрес.

Ссылка может присвоить "пустое" значение null, не указывающее ни на какой адрес оперативной памяти:

ar = null;

После этого массив, на который указывала данная ссылка, теряется, если на него не было других ссылок.

Кроме простой операции присваивания, со ссылками можно производить еще только сравнения на равенство, например, а==b, и неравенство, а!=b. При этом сопоставляются адреса, содержащиеся в ссылках, мы можем узнать, не ссылаются ли они на один и тот же массив.

Кроме ссылки на массив, для каждого массива автоматически определяется целая константа с одним и тем же именем length. Она равна длине массива. Для каждого массива имя этой константы уточняется именем массива через точку. Так, после наших определений, константа a.length равна 5, константа b.length равна 100, a ar.length равна 50.

Последний элемент массива а можно записать так: a[a.length-1], предпоследний – a[a.length-2] и т. д. Элементы массива обычно перебираются в цикле вида:

double aMin = a[0], aMax = aMin;

for (int i = 1; i < a.length; i++){

if (a[i] < aMin) aMin = a[i];

if (a[i] > aMax) aMax = a[i];

}

**Консольный ввод-вывод данных**

Для вывода данных можно использовать конструкцию типа:

System.out.println(данные);

При этом нужно помнить, что данные соединяются знаком +, например, блок кода

int n=5;

System.out.println("Задача"+" "+"#"+n);

выведет на экран

Задача #5

и переведет курсор на следующую строчку.

Для вывода пустой строчки используется та же конструкция, но без параметров.

Если перевод строки не нужен, то можно использовать конструкцию типа

System.out.print(данные);

Для ввода данных с клавиатуры необходима более сложная конструкция:

Scanner in = new Scanner(System.in);

int m;

m = in.nextInt();

В первой строчке создается объект in класса Scanner – специального класса для ввода данных, причем в конструкторе класса указывается потока ввода System.in, означающий, что ввод будет с клавиатуры. Далее, объявляется переменная целого типа, которой потом присваивается целое значение, введенное в поток in (in.nextInt()). Для работы данной инструкции необходимо подключить класс java.util.Scanner (import java.util.Scanner;). Вводить можно разные типы данных, используя соответствующий метод (nextInt, nextShort, nextFloat, nextDouble и т.д., для всех примитивных типов). Например, программа, считывающая с клавиатуры массив и вычисляющая среднеарифметическое из его элементов:

import java.util.Scanner;

public class hi {

public static void main (String [] args){

int n, mas[];

float s=0;

Scanner in = new Scanner(System.in);

System.out.print("Введите размерность массива: ");

n = in.nextInt();

mas = new int[n];

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

System.out.print("Введите "+i+"-й"+" элемент: ");

mas[i]=in.nextInt();

s+=mas[i];

}

s/=n;

System.out.println("Ср. арифм. массива: "+s);

}

}

После запуска программы увидим:

Введите размерность массива: 3

Введите 0-й элемент: 3

Введите 1-й элемент: 2

Введите 2-й элемент: 2

Ср. арифм. массива: 2.3333333

По умолчанию компилятор выводит 7 знаков после запятой для вещественных чисел. Если нужно вывести меньше, то нужно слегка подкорректировать наш код:

import java.util.Scanner;

import java.text.NumberFormat;//класс для форматирования представления //чисел

public class hi {

public static void main (String [] args){

int n, mas[];

float s=0;

Scanner in = new Scanner(System.in);

System.out.print("Введите размерность массива: ");

n = in.nextInt();

mas = new int[n];

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

System.out.print("Введите "+i+"-й"+" элемент: ");

mas[i]=in.nextInt();

s+=mas[i];

}

s/=n;

NumberFormat nf = NumberFormat.getInstance(); //создаем объект //nf класса NumberFormat и приравниваем его к установленному в //системе формату представления чисел //(NumberFormat.getInstance())

nf.setMaximumFractionDigits(2);//устанавливаем максимальное //количество цифр после запятой равным 2

System.out.println("Ср. арифм. массива: "+nf.format(s));

//выводим переменную s с использованием установленного формата

}

}

После запуска получим:

Введите размерность массива: 3

Введите 0-й элемент: 3

Введите 1-й элемент: 2

Введите 2-й элемент: 2

Ср. арифм. массива: 2,33

Но массивы удобнее заполнять с использованием генератора случайных чисел. Для этих целей в языке Java есть класс Random. Поэтому в программе можно создать объект данного класса и использовать его метод nextInt(*верхняя граница*), выдающий случайное число из диапазона 0..(*верхняя граница* *–* *1).* Для работы генератора нужно подключить java.util.Random. С учетом описанного наш пример можно переписать следующим образом:

import java.text.NumberFormat;

import java.util.\*; //подключаем весь пакет java.util

public class hi {

public static void main (String [] args){

int n, mas[];

float s=0;

Scanner in = new Scanner(System.in);

System.out.print("Введите размерность массива: ");

n = in.nextInt();

mas = new int[n];

Random random = new Random();//создаем объект-//генератор случ. чисел

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

mas[i]=random.nextInt(5);//генерируем очередное случ. //число из нужного нам диапазона (от 0 до 4)

System.out.println(i+"-й"+" элемент: "+mas[i]);

s+=mas[i];

}

s/=n;

NumberFormat nf = NumberFormat.getInstance();

nf.setMaximumFractionDigits(2);

System.out.println("Ср. арифм. массива: "+nf.format(s));

}

}

После запуска получим:

Введите размерность массива: 10

0-й элемент: 4

1-й элемент: 3

2-й элемент: 4

3-й элемент: 2

4-й элемент: 3

5-й элемент: 4

6-й элемент: 2

7-й элемент: 3

8-й элемент: 1

9-й элемент: 1

Ср. арифм. массива: 2,7

Задачи:

1. **Одномерные массивы.**
2. Дан массив. Удалить из него нули и после каждого числа, оканчивающего на 5, вставить 1.
3. Случайным образом генерируется массив чисел. Пользователь вводит числа a и b. Заменить элемент массива на сумму его соседей, если элемент массива четный и номер его лежит в промежутке от a до b.
4. В одномерном массиве удалить промежуток элементов от максимального до минимального.
5. Дан одномерный массив. Переставить элементы массива задом-наперед.
6. Сформировать одномерный массив случайным образом. Определить количество четных элементов массива, стоящих на четных местах.
7. Задается массив. Определить порядковые номера элементов массива, значения которых содержат последнюю цифру первого элемента массива 2 раза (т.е. в массиве должны быть не только однозначные числа).
8. Сформировать одномерный массив из целых чисел. Вывести на экран индексы тех элементов, которые кратны трем и пяти.
9. Задается массив. Написать программу, которая вычисляет, сколько раз введенная с клавиатуры цифра встречается в массиве.
10. Задается массив. Узнать, какие элементы встречаются в массиве больше одного раза.
11. Даны целые числа а1, а2,..., аn. Вывести на печать только те числа, для которых аi ≥ i.
12. Дан целочисленный массив с количеством элементов n. Напечатать те его элементы, индексы которых являются степенями двойки.
13. Задана последовательность из N чисел. Определить, сколько среди них чисел меньших К, равных К и больших К.
14. Задан массив действительных чисел. Определить, сколько раз меняется знак в данной последовательности чисел, напечатать номера позиций, в которых происходит смена знака.
15. Задана последовательность N чисел. Вычислить сумму чисел, порядковые номера которых являются простыми числами.
16. Дан массив чисел. Указать те его элементы, которые принадлежат отрезку [с, d].
17. Массив состоит из нулей и единиц. Поставить в начало массива нули, а затем единицы.
18. Дан массив целые положительных чисел. Найти среди них те, которые являются квадратами некоторого числа х.
19. В массиве целых чисел найти наиболее часто встречающееся число. Если таких чисел несколько, то определить наименьшее из них.
20. Дан целочисленный массив с количеством элементов n. Сжать массив, выбросив из него каждый второй элемент.
21. Дан массив, состоящий из n натуральных чисел. Образовать новый массив, элементами которого будут элементы исходного, оканчивающиеся на цифру k.
22. Даны действительное число х и массив A[n]. В массиве найти два члена, среднее арифметическое которых ближе всего к х.
23. Даны два массива А и В. Найти, сколько элементов массива А совпадает с элементами массива В.
24. **Двумерные массивы.**
    1. Написать программу, генерирующую магические квадраты заданного пользователем размера.
    2. Дан двумерный числовой массив. Значения элементов главной диагонали возвести в квадрат.
    3. Дан двумерный массив. Поменять местами значения элементов столбца и строки на месте стыка минимального значения массива (или первого из минимальных). Например, если индекс минимального элемента (3;1), т.е. он находится на пересечении 3 строки и 1 столбца, то 3 строку сделать 1 столбцом, а 1 столбец сделать 3 строкой.
    4. Дан двумерный массив. Сформировать одномерный массив только из четных элементов двумерного массива.
    5. Дан двумерный массив. Найти сумму элементов массива, начиная с элемента, индексы которого вводит пользователь, и заканчивая элементом, индексы которого вводит пользователь.
    6. Дан двумерный массив. Сформировать одномерный массив только из элементов двумерного массива с четной суммой индексов.
    7. Дан двумерный массив. Сделать из него 2 одномерных: в одном – четные элементы двумерного массива, в другом – нечетные.
    8. Вычислить сумму и число положительных элементов матрицы A[N, N], находящихся над главной диагональю.
    9. В квадратной матрице определить максимальный и минимальные элементы. Если таких элементов несколько, то максимальный определяется по наибольшей сумме своих индексов, минимальный – по наименьшей.
    10. Для заданной квадратной матрицы сформировать одномерный массив из ее диагональных элементов.
    11. Заданы матрица порядка n и число k. Вычесть из элементов k-й строки диагональный элемент, расположенный в этой строке.
    12. Заданы матрица порядка n и число k. Вычесть из элементов k-го столбца диагональный элемент, расположенный в этом столбце.
    13. Дана прямоугольная матрица. Найти строку с наибольшей и наименьшей суммой элементов. Вывести на печать найденные строки и суммы их элементов.
    14. Дана прямоугольная матрица. Найти столбец с наибольшей и наименьшей суммой элементов. Вывести на печать найденные столбцы и суммы их элементов.
    15. Дан двумерный массив. Выяснить, в каких строках сумма элементов меньше введенного пользователем значения.
    16. Дан двумерный массив. Выяснить, в каких столбцах произведение элементов меньше введенного пользователем значения.
    17. Дан двумерный массив. Выяснить, есть ли столбец и строка с одинаковой суммой элементов. Если есть, напечатать их номера.
    18. Дан двумерный массив. Выяснить, есть ли столбец и строка с одинаковым произведением элементов. Если есть, напечатать их номера.
    19. Дан двумерный массив. Выяснить, есть ли строки с одинаковой суммой элементов. Если есть, вывести их номера.
    20. Дан двумерный массив. Выяснить, есть ли столбцы с одинаковой суммой элементов. Если есть, вывести их номера.
    21. Дан двумерный массив. Определить максимальный среди положительных элементов, минимальный среди отрицательных элементов и поменять их местами.
    22. Дан двумерный массив. Заменить первый нуль в каждом столбце на количество нулей в этом столбце.
    23. Дан двумерный массив. Заменить первый нуль в каждой строке на количество нулей в этой строке.