2.1. Строки

Строки – это последовательности символов, заключенные в кавычки. Символы внутри строк можно представлять их знаками или эскейп-последовательностями, например,

“abc\tABC\n123\0101\0102”

.Строка символов должна быть расположена в пределах одной строки программы.

2.2. Переменные

Переменные содержат данные, с которыми работает программа. Объявление переменной имеет вид

```
int i, j, k;    // Три переменные целого типа
double x = 1.0; // Инициализированная переменная
```

Возможна динамическая инициализация создаваемых переменных, когда для инициализации используется выражением, вычисляемое при их создании.

```
// Демонстрирует динамическую инициализацию.
class DynInit {
   public static void main(String args[]) {
      double a = 3.0, b = 4.0;
      // Переменная c динамически инициализирована
      double c = Math.sqrt(a * a + b * b);
      System.out.println("Гипотенуза равна " + c);
   }
}
```

Для вычисления гипотенузы вызывается встроенный метод `sqrt` класса `Math` извлечения квадратного корня.

2.3. Область действия и время жизни переменных

Переменные можно объявлять в любом блоке программы, ограниченном фигурными скобками. Переменная видна от точки объявления до конца блока, в том числе и во вложенных блоках. Не допускается иметь переменные с одинаковыми именами во внешнем и во вложенном блоке. В C++ это разрешается.

```
{
    int x = 1;
    ...
    {
        int x = 2;  // Нельзя
        ...
    }
}
```

Переменная создается при входе в блок и уничтожается при выходе из блока. Если переменная инициализируется при создании, то инициализация повторяется при каждом входе в блок.

2.4. Преобразование и приведение типов

Если типы совместимы, то Java выполняет автоматическое преобразование, например, значение типа `int` можно присвоить переменной типа `long`. Не определено никакого преобразования из `double` в `byte`, то есть эти типы несовместимы.

Чтобы сделать преобразование между несовместимыми типами, следует использовать операцию приведения типов.

Автоматическое преобразование типов

При присваивании переменной одного типа значения другого типа происходит автоматическое преобразование при выполнении условий

1. два типа совместимы;
2. целевой тип больше исходного.

Например, тип `int` больше чем `byte`, так как содержит все значения `byte`, поэтому преобразование из `byte` в `int` происходит автоматически.

Числовые типы несовместимы с `char` и `boolean`. Эти типы не совместимы между собой.

Приведение несовместимых типов

Приведение имеет формат:

```
(target-type) value
```

Здесь `(target-type)` определяет желаемый тип, к которому следует преобразовать значение `value`. Вот пример приведения:

```
int a;
byte b;
…
b = (byte) a;
```

Если значение `a` окажется больше диапазона `byte`-типа (256), то оно будет редуцировано до остатка от деления на 256.

При приведении числа с плавающей точкой к целому, дробная часть отбрасывается.

Автоматическое расширение типа в выражениях

При вычислениях с целыми типами значения типа `byte` и `short` преобразуются в `int`. Это может приводить к ошибкам, например,

```
byte b = 50;
b = b * 2;      // Ошибка - присваивание byte-переменной int-значения
```

Последнюю строку следует писать в виде:

```
b = (byte)(b * 2);
```

2.5. Массивы

Массивы – это наборы однотипных элементов, к которым можно обращаться по номеру. Массивы могут иметь несколько измерений.

Одномерные массивы

Для создания массива выполняются два шага. Сначала создается переменная типа массив, например,

```
int month_days[];
```

Это объявление говорит, что `month_days` является массивом, хотя сразу после объявления эту переменную использовать нельзя, так как предварительно надо создать массив в памяти оператором `new`:

```
month_days = new int[12];
```

Теперь массив из 12 элементов создан в памяти. Все элементы массива при создании инициализируются нулями.

Таким образом, в Java все массивы являются динамическими.

Можно совместить объявление массива и выделение памяти:

```
int month_days[] = new int[12];
```

Массивы можно инициализировать при объявлении:

```
int month_days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

Инициализаторы заключаются в фигурные скобки. Оператор new не нужен. Размер массива равен числу инициализаторов, в приведенном примере это 12.

Нумерация элементов массива начинается с нуля.

При выполнении программы проверяется, чтобы индекс массива не вышел за установленные пределы. Для массива month_days индекс может изменяться от 0 до 11 включительно. При выходе индекса за границы массива возникает ошибка времени выполнения.

Программа 5. Определение числа уникальных элементов массива

В программе создается массив, заполняется случайными числами и подсчитывается число разных элементов массива.

```
import java.io.*;
import java.util.Random;     // Для случайных чисел
import java.util.Scanner;    // Для ввода целых

public class UnicElemInArray {
   public static void main(String args[])
   throws IOException
   {
      int ArraySize;     // Размер массива

      System.out.println("Введите размер массива");
      Scanner rdr = new Scanner(System.in);    // Объект для чтения с клавиатуры

      ArraySize = rdr.nextInt();  // Ввод размера массива.
                                  // Метод nextInt() вводит с клавиатуры целое

      int x[] = new int[ArraySize];
      int i;
      // Заполнение массива случайными числами
      Random rnd = new Random();  // Создание генератора случайных чисел
      for(i = 0; i < ArraySize; i++){
         x[i] = rnd.nextInt() % 10; // Получение следующего случайного числа
         System.out.print(x[i] + "\t");
      }
      System.out.println();
      // Определяем, сколько в массиве разных элементов
      int f[] = new int[ArraySize];  // Массив признаков 1 - уникальное, 0 - не уникальное
      for(i = 0; i < ArraySize; i++)
         f[i] = 1;                   // Предполагаем, все элементы массива - уникальные
      int j = 0;
      while(j < ArraySize - 1){
      // Ищем уникальное число
         while(j < ArraySize - 1 && f[j] == 0)
            j++;
         if(j == ArraySize - 1)   // Дошли до конца массива
            break;                // Выход из цикла
         // Просмотр массива, начиная с номера j + 1, фиксация совпадений
         for(i = j + 1; i < ArraySize; i++)
            if(x[i] == x[j])
               f[i] = 0;
         j++;
      }
      for(i = 0; i < ArraySize; i++)
         System.out.print(f[i] + "\t");
      // Подсчет разных элементов
```

```
      int unic = 0;
      for(i = 0; i < ArraySize; i++)
         unic += f[i];
      System.out.println();
      System.out.println("В массиве " + unic + " различных элементов");


   }
}
```

Пример вывода программы:

```
Введите размер массива
20
2  -9 -1 0  6  -4 -2 1  9  -8 8  8  8  6  -7 -6 7  7  1  8
1  1  1  1  1  1  1  1  1  1  1  0  0  0  1  1  1  0  0  0
В массиве 14 различных элементов
```

2.6. Многомерные массивы

Многомерные массивы являются, по сути дела, массивами массивов. Каждый индекс многомерного массива помещается в отдельные квадратные скобки. Следующая инструкция определяет двумерный массив 4×5.

```
int a[][] = new int[4][5];
```

В массиве a 4 строки и 5 столбцов. Каждая строка – это массив из 5 элементов типа int. Поскольку каждая строка – это массив, размеры строк могут быть разными. Пример массива, в котором строки имеют разную длину, приведен в программе 6. Конечно, чаще всего встречаются обычные двумерные прямоугольные массивы, используемые для моделирования матриц.

Программа 6. Треугольный массив

В следующей программе создается «треугольный» массив, в котором длины строк возрастают от 1 до 5. Элементам массива присваиваются последовательные целые числа. Затем массив выводится.

```
import java.io.*;
public class TriangularArray {
   public static void main(String args[])
   {
      int  n = 5;                         // Число строк массива
      int trar[][] = new int[n][];        // "Заготовки" для n строк
      // Создание строк
      for(int i = 0; i < n; i++)
         trar[i] = new int[i + 1];        // Создание строк длиной i элементов
      // Заполнение строк
      int i, j, k = 0;
      for(i = 0; i < n; i++)              // Перебор строк
         for(j = 0; j < i + 1; j++){      // Перебор элементов строк
            k++;
            trar[i][j] = k;
         }
   // Вывод массива
      for(i = 0; i < n; i++){
         for(j = 0; j < i + 1; j++)
            System.out.print(trar[i][j] + "\t");
         System.out.println();
      }
   }
}
```

Вывод данной программы имеет следующий вид:

```
1
2  3
4  5  6
7  8  9  10
11 12 13 14 15
```

Многомерные массивы можно инициализировать при определении. Указывая инициализаторы в фигурных скобках, например,

```
int a[][] =
{
   {1, 1 + 1, 1 + 2, 4},
   {5, 6, 7, 8},
   {9, 10, 11,12}
};
```

Размерность массива определяется по списку инициализации, для приведенного примера это 3×4.

Есть альтернативная возможность расположения квадратных скобок при объявлении массива. Следующие инструкции эквивалентны:

```
int a1[] = new int[5];
int [] a2 = new int[5];
```

Многомерные массивы также можно определять двумя эквивалентными способами:

```
double b1[][] = new double [3][4];
double [][]b2 = new double [3][4];
```

Задачи 5 - 11. Массивы

5. Напишите программу, переставляющую элементы массива в обратном порядке.
6. Напишите программу, определяющую число четных и нечетных элементов в массиве.
7. Напишите программу сортировки массива методом пузырька.
8. Переписать программу 5 по определению числа уникальных элементов массива, используя следующий алгоритм. Завести счетчик. Перебирать элементы массива и каждый сравнивать с предшествующими. Если будут совпадения, значит рассматриваемое число уже было в массиве. Иначе увеличить счетчик.
9. Напишите программу, в которой создается «квадратный» массив, заполняется, например, последовательными целыми числами, транспонируется (элементы строк и столбцов меняются местами) и выводится.
10. Напишите программу, создающую в памяти «треугольный» массив, содержащий элементы треугольника Паскаля и выводящую его.
11. Напишите программу, отыскивающую седловые точки прямоугольной матрицы, то есть элементы, максимальные в своей строке и, одновременно, минимальные в своем столбце или минимальные в своей строке и максимальные в своем столбце. Выводите номера строк и столбцов седловых точек и их значение.