

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования «Севастопольский государственный университет»

Институт информационных технологий и управления в технических системах  
Кафедра информационных технологий и компьютерных систем

ПРОГРАММИРОВАНИЕ. БАЗОВЫЕ ПРОЦЕДУРЫ ОБРАБОТКИ  
ИНФОРМАЦИИ

Часть 3

Обработка одномерных массивов и матриц

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по дисциплине

«Программирование. Базовые процедуры обработки информации»

для студентов направлений

09.03.01 – Информатика и вычислительная техника и

09.03.04 – Программная инженерия

дневной формы обучения

Севастополь  
2020

УДК 004.42(076.5)

ББК 32.973-018я7

П 784

**П 784** Программирование. Базовые процедуры обработки информации. Методические указания к лабораторным работам по дисциплине «Программирование. Базовые процедуры обработки информации» для студентов направлений 09.03.01 – Информатика и вычислительная техника и 09.03.04 – Программная инженерия дневной формы обучения. [В 4 частях]. Часть 3. Обработка одномерных массивов и матриц / Министерство образования и науки Российской Федерации, ФГАОУ ВО «Севастопольский государственный университет», Институт информационных технологий и управления в технических системах, кафедра информационных технологий и компьютерных систем ; сост. Т. В. Волкова, Е. С. Владимирова. – Севастополь : [Изд-во СевГУ], 2020. – 32 с. – Текст : непосредственный

Целью методических указаний является оказание помощи в подготовке к лабораторным работам по дисциплине «Программирование. Базовые процедуры обработки информации», предусматривающим разработку и выполнение программ обработки одномерных и двумерных массивов с использованием методологии функционального программирования. Методические указания предназначены для студентов первого курса дневной формы обучения направлений 09.03.01 – Информатика и вычислительная техника и 09.03.04 – «Программная инженерия».

УДК 004.42(076.5)

ББК 32.973-018я7

Методические указания рассмотрены и утверждены на заседании кафедры информационных технологий и компьютерных систем 06.11.2019 г., протокол № 3.

Рецензент: докт. техн. наук, профессор кафедры информационных систем Ю.В. Доронина.

Допущено учебно-методическим центром СевГУ в качестве методических указаний.

## СОДЕРЖАНИЕ

1. ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ .....	4
2. ЛАБОРАТОРНАЯ РАБОТА № 8. ИССЛЕДОВАНИЕ АЛГОРИТМОВ ОБРАБОТКИ ОДНОМЕРНЫХ МАССИВОВ .....	6
3. ЛАБОРАТОРНАЯ РАБОТА № 9. ИССЛЕДОВАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ ОПЕРАЦИИ НАД МАТРИЦАМИ.....	21
ПЕРЕЧЕНЬ ССЫЛОК.....	31

## 1. ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Для освоения основ программирования на языке Java предлагается серия лабораторных работ. Каждая работа требует изучения методических указаний и рекомендованной литературы, подготовку текста программы, компиляцию программы некоторое ее исследование.

Все действия, указанные в методических указаниях к отдельной работе (освоение практических приемов использования системы разработки программ, объяснение примененных элементов языка Java, ответы на вопросы, задания) обязательно должны быть выполнены до следующего лабораторного занятия.

По выполненной лабораторной работе составляется отчет. Оформление отчета должно соответствовать требованиям к текстовым учебным документам соответствующих ГОСТов [Электронный фонд правовой и нормативно-технической документации. <http://docs.cntd.ru>].

Отчет представляется на листах формата А4 и в электронном виде (файл Microsoft Word). Первый лист отчета – титульный. На нем указывается название вуза, название выпускающей кафедры, название работы, ФИО и группа исполнителя, ФИО принимающего работу, город (Севастополь) и год. Пример оформления титульного листа приведен в приложении А. Следующие листы являются собственно отчетом и должны содержать следующие разделы:

- 1) цель работы
- 2) постановка задачи;
- 3) анализ задачи, выявляющий связи между элементами задачи (обоснование типов входных и выходных данных, описание реализуемых функций);
- 4) схема и описание алгоритма решения задачи;
- 5) тестовые примеры и результаты их обработки вручную;
- 6) текст Java-программы, заданной вариантом задания;
- 7) сведения об отладке программы и проверке ее работоспособности (описание ошибок (синтаксических и логических), выявленных на этапе отладки программы, результаты работы (в виде скриншотов), сравнение результатов работы программы на тестовых примерах с результатами ручных расчетов);
- 8) выводы (констатирует решение всех задач, описанных в разделе «Постановка задачи»).

Студент обязан завести рабочую тетрадь, в которой должны фиксироваться инструкции к работе, вопросы, возникающие при выполнении работы и ответы на них, черновики и фрагменты программ. В рабочей тетради можно представлять отчеты по работам. При этом титульный лист каждого отчета должен находиться на правой странице разворота. Рабочую тетрадь обязательно иметь на каждом лабораторном занятии – будут оцениваться качество и полнота выполненных заданий.

Студент допускается к защите отчета по лабораторной работе после того, как он продемонстрирует преподавателю выполнение поставленной задачи на компьютере (результаты работы программы и/или другое задание, например, работу с окном кода BlueJ) и предоставит папку с разработанным java-проектом (в электронном виде). Рекомендуемое имя папки: Лаб\_n\_m\_Фамилия\_Группа\_Год, где n – номер лабораторной работы, m – номер проекта (используется если задание по лабораторной работе предусматривает разработку нескольких проектов). Текстовый файл с отчетом по работе рекомендуется поместить в папку проекта.

Защита лабораторных работ состоит из доклада студента о проделанной работе с объяснением содержания отчета. Студент должен ответить на контрольные вопросы к соответствующей лабораторной работе, приведенные в методических указаниях, а также другие вопросы преподавателя, касающиеся поставленной задачи, показать работоспособность подготовленной программы и навыки работы с программной системой. При защите текущей работы возможны вопросы по темам предыдущих лабораторных работ. Результат защиты оценивается по шкале 0 – 100 баллов (ESTC).

Для подготовки программ студентам рекомендуется установить на своих компьютерах последнюю версию системы разработки программ BlueJ и соответствующую версию комплекта разработчика Java Development Kit (<http://www.bluej.org>).

Выполнив несложную настройку BlueJ, можно выбрать наиболее удобный для Вас язык интерфейса.

Рекомендуется иметь съемный носитель информации (флэш-диск), на котором будут храниться проекты лабораторных работ и соответствующие отчеты.

## 2. ЛАБОРАТОРНАЯ РАБОТА № 8. ИССЛЕДОВАНИЕ АЛГОРИТМОВ ОБРАБОТКИ ОДНОМЕРНЫХ МАССИВОВ

### 2.1. Цель работы

Целью данной работы является исследование циклических алгоритмов и программ, осуществляющих типичные операции над одномерными массивами, получение навыков разработки и использования подпрограмм (методов-процедур и методов-функций), а также изучение возможностей отладчика BlueJ.

### 2.2. Постановка задачи

Разработать программу, реализующую обработку массива данных по заданию, указанному в таблице 2.1, в соответствии с номером варианта. Программа должна удовлетворять нижеперечисленным требованиям.

Основная программа (метод `main`) должна вызывать четыре вспомогательные подпрограммы (четыре метода).

Первый метод-функция предназначен для вычисления значения в соответствии с заданием, указанным в столбце 3 таблицы вариантов.

Второй метод-процедура предназначен для выполнения сортировки элементов массива, метод и тип которой заданы столбцами 4 и 5 таблицы вариантов.

Третий метод-процедура предназначен для обработки массива способом, указанным в столбце 6 таблицы вариантов.

Четвертый метод-процедура предназначен для вывода в окно терминала всех элементов массива. Элементы массива должны быть выведены по  $n$  элементов в строке, где  $n$  – параметр метода.

Значения переменных  $A$ ,  $B$ ,  $m$  (в зависимости от варианта) должны быть параметрами методов и задаваться в основной программе.

Основная программа (метод `main`) должна выполнять следующие действия:

1) инициализировать два различных массива (отличаются значения элементов и длины массивов); базовый тип массива (тип элементов) определен вариантом;

2) для каждого из массивов вывести:

исходный массив (метод 4);

результат выполнения метода-функции для исходного массива (метод 1);

результат сортировки исходного массива (метод 2, метод 4);

результат выполнения метода-процедуры для исходного (не отсортированного) массива (метод 3, метод 4).

### 2.3. Внеаудиторная подготовка

Для подготовки к лабораторной работе следует изучить краткие теоретические сведения, приведенные в пункте 2.4 методических указаний. Рекомендуется также ознакомиться с [1] (с.72-79) и [2].

## 2.4. Краткие теоретические сведения

### 2.4.1. Одномерные массивы

**Одномерный массив** в Java – это тип данных, имеющий две части. Одна часть – контейнер, в котором содержатся однотипные элементы. Другая часть – поле `length`, в котором хранится количество элементов первой части [3]. Поле `length` получает значение при создании массива и потом не изменяется. На рисунке 2.1 изображен массив из 10 элементов.

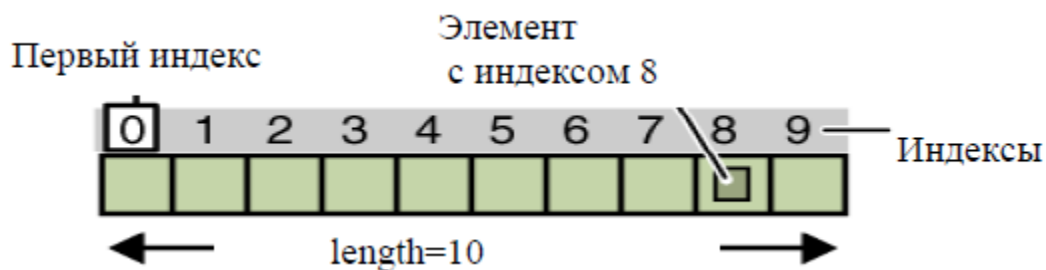


Рисунок 2.1. Массив из десяти элементов

Для выбора элемента нужно задать его числовой индекс, т.е. номер. Элементы нумеруются с 0. В массиве из десяти элементов индекс последнего элемента равен 9.

**Объявление массива** – это объявление переменной, в которой хранится ссылка на массив. Примеры (в панели кода BlueJ следует использовать формы из левого столбца):

<code>int[ ] МассивЦелых;</code>	или	<code>int МассивЦелых[ ];</code>
<code>byte[ ] МассивБайтов;</code>	или	<code>byte МассивБайтов[ ];</code>
...		
<code>double[ ] МассивДвТочности;</code>	или	<code>double МассивДвТочности[ ];</code>
<code>char[ ] МассивСимволов;</code>	или	<code>char МассивСимволов[ ];</code>
<code>String[ ] МассивСтрок;</code>	или	<code>String МассивСтрок[ ];</code>

В объявлении массива `A (int[ ] A;)` тип переменной `A` – это `int[ ]` (массив целых чисел), а тип элементов массива (**базовый тип**) – это `int`. При **объявлении** массива память для элементов не выделяется, создается только ссылка (`A`) с начальным значением `null`.

Чтобы после объявления переменная (`A`) действительно представляла массив, она должна иметь значение, отличное от `null`.

**Создание массива** – это выделение памяти для хранения указанного количества элементов и заполнение поля `length` этого массива. Переменная – ссылка (A) получает при создании массива отличное от `null` значение.

Обычно выделение памяти (в процессе выполнения программы) выполняет оператор `new`:

```
int[ ] A; //объявлена ссылка A на массив
A = new int[10]; //выделена память для десяти int, A.length=10
```

Можно совместить объявление переменной-ссылки и создание массива:

```
int[ ] A = new int [5];
```

Другой вариант создания массива – **инициализация**, т.е. использование массива – литерала:

```
int[ ] B = {-5, 29, 0, -45, 55}; //создается массив из пяти элементов,
//указанных в фигурных скобках.
```

При выполнении этого оператора в оперативной памяти будет создан массив из 5 элементов типа `int`, которым будут присвоены заданные литералом значения. Адрес этого массива будет записан в переменную B (рисунок 2.2). Ссылка на массив (значение адреса массива) на рисунке обозначена «жирной» точкой и стрелкой. Пустая ссылка (`null`) обычно изображается «жирной» точкой без стрелки. Например, если в программе есть объявление массива `int [ ] X;`, то пока он не инициализирован, или не создан процедурой `new`, в ячейке ОП с адресом X будет храниться значение `null` (рисунок 2.2).

Кроме того, ссылку можно сделать равной другой ссылке на массив (рисунок 2.3):

```
int[ ] C={1, 2, 3}; // создан массив C.
int[ ] D; // объявлен массив D.
D = C; // переменная D стала синонимом переменной C, т.е.
// D ссылается на тот же массив, что и C.
```

**Операция отношения** `D==C` вернет `true`, т.к. ссылки (адреса массивов) C и D равны. Обратите внимание, что в данном случае сравниваются ссылки, а не содержимое массивов. Фактически, операцию сравнения `D==C`, можно трактовать, как проверку «C и D – это один и тот же массив?».



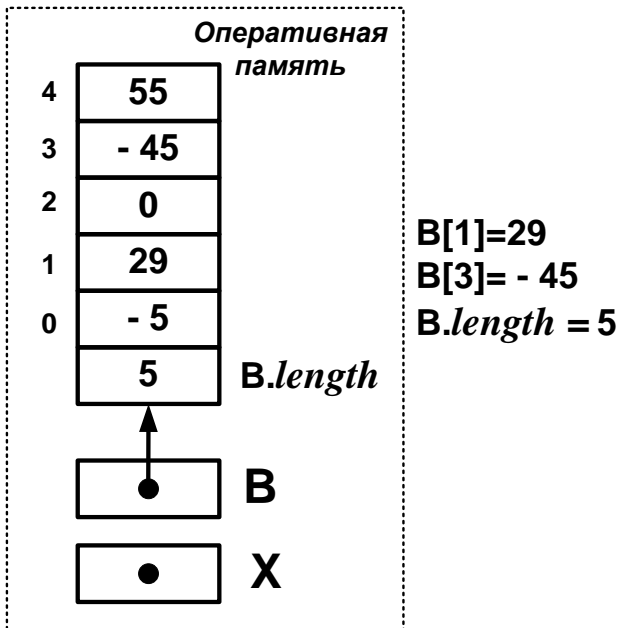


Рисунок 2.2 – Массив целых чисел B и пустая ссылка на массив целых чисел X

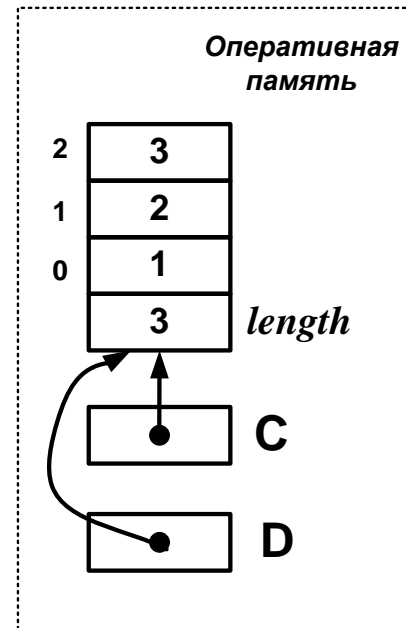


Рисунок 2.3 – Ссылки на один и тот же массив – синонимы C и D

**Заполнение массива** проводится заданием значений отдельных элементов. Часто используют цикл, в теле которого задается значение одного очередного элемента. При отладке программ можно массив заполнять числами, зависящими от номера элемента или случайными числами при помощи метода `java.lang.Math.random()`, дающего действительное псевдослучайное число из диапазона 0..1.

#### **Типовые задачи для одномерных массивов:**

- вычисление суммы элементов массива;
- подсчет количества элементов массива, удовлетворяющих данному условию;
- нахождение элемента с экстремальным (минимум, максимум) значением;
- копирование массива;
- сортировка элементов (расположение в порядке возрастания или убывания);
- поиск данного элемента в массиве.

Методика решения первых трех задач одинакова: исходя из текущего значения (суммы, количества, минимума, максимума), скорректировать его, рассматривая очередной элемент массива. После выполнения коррекции с учетом всех элементов массива будут найдены искомые сумма, количество элементов, минимум или максимум.

Правила коррекции:

- сумма: текущее значение суммы увеличить на значение очередного элемента;
- количество: если очередной элемент «подходит», увеличить количество на 1;
- минимум: если очередной элемент меньше текущего минимума, заменить минимум значением этого элемента;
- максимум: если очередной элемент больше текущего максимума, заменить максимум значением этого элемента.

При поиске экстремальных значений важен выбор начального значения экстремума. В качестве такового следует выбрать значение одного из элементов массива, например,  $A[0]$ . При вычислении суммы и количества элементов начальными значениями должны быть нули.

Копирование массива производится поэлементно. Класс `System` содержит метод `arraycopy()`, предоставляющий различные варианты копирования массива.

Класс `Arrays` содержит методы для выполнения сортировки (`sort`) и поиска (`binarySearch`), а также методы для копирования и заполнения массивов.

В качестве примера программы обработки одномерных массивов на рисунке 2.4. приведен класс `minElDemo`, содержащий методы `minElArr` (возвращает значение минимального элемента массива целых чисел, переданного в качестве параметра), `putArr` (выводит элементы массива в одной строке), `main` (вызывает методы `minElArr` и `putArr` для обработки массивов  $A$  и  $B$ ).

#### 2.4.2. Работа с отладчиком BlueJ

Выполнение программы можно контролировать, используя отладчик BlueJ. Отладчик позволяет наблюдать за значениями переменных и выполнять программу в медленном темпе, по шагам. При этом в тексте программы отмечается строка, готовая для выполнения, а в окне отладчика видны текущие значения переменных. Работа с отладчиком продемонстрирована на рисунках 2.4-2.8.

Для включения в работу отладчика поставьте в программе точку останова (щелкните левой клавишей мыши на номере нужной строки кода) – рисунок 2.4. После запуска метода `main` программа остановится в точке останова (рисунок 2.5). Двойной щелчок левой клавишей мыши по строке

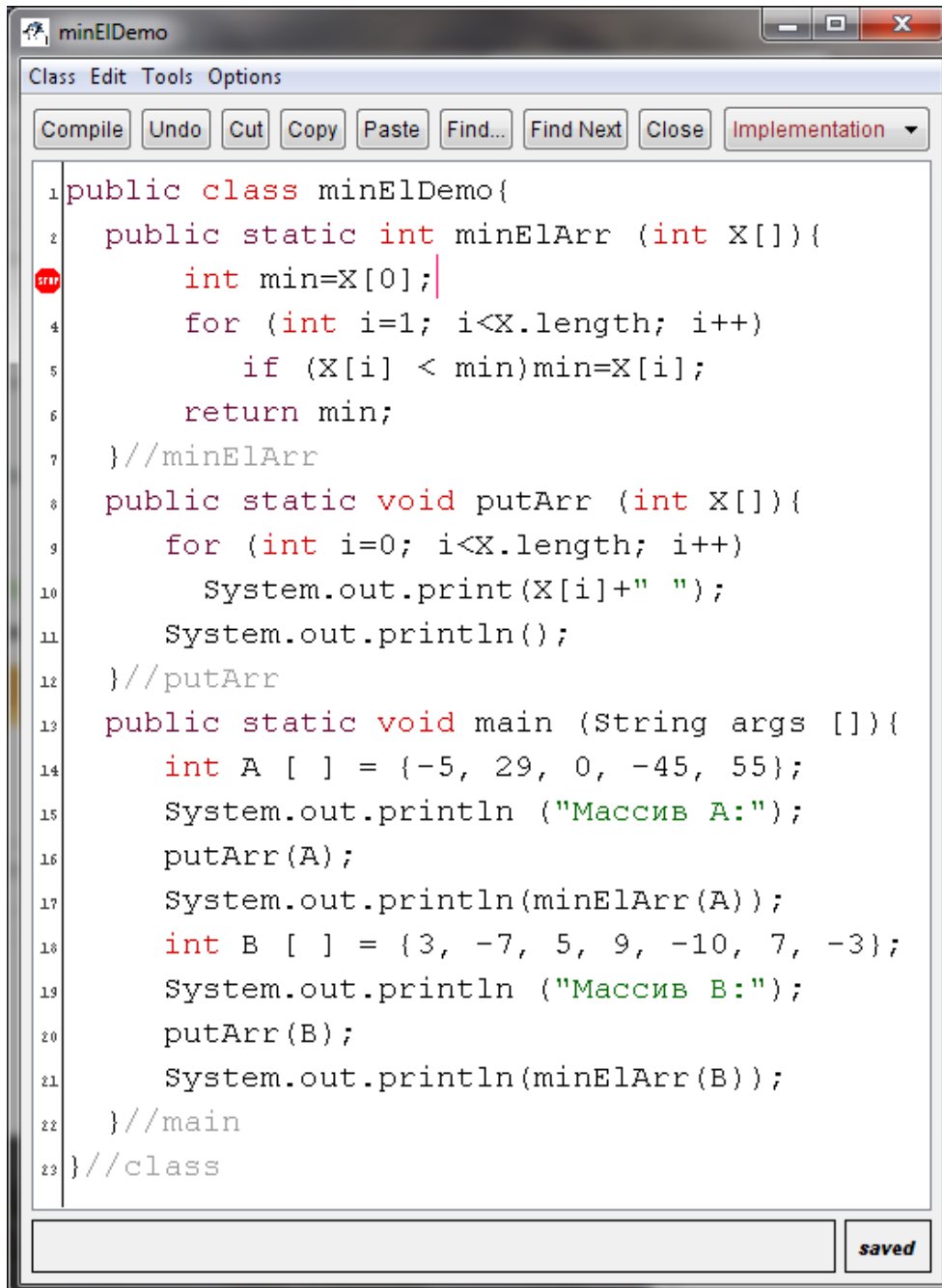
`int[]X=<object reference>` в разделе `Local variables`

позволяет вывести значения элементов массива (рисунок 2.6).

На рисунках 2.7 и 2.8 изображены результаты пошагового выполнения программы для двух шагов. Шаг – одно нажатие на кнопку `<Step>`.

Кнопка `<Continue>` продвигает процесс выполнения сразу до следующей точки останова. Чтобы закончить отладку, надо нажать на кнопку `<Terminate>` и закрыть окно отладчика.

Если в процессе отладки изображение отладчика исчезло, можно вернуть его на экран, выбрав пункт меню View → Show Debugger.



```
1 public class minElDemo{
2     public static int minElArr (int X[]){
3         int min=X[0];
4         for (int i=1; i<X.length; i++)
5             if (X[i] < min)min=X[i];
6         return min;
7     }//minElArr
8     public static void putArr (int X[]){
9         for (int i=0; i<X.length; i++)
10            System.out.print(X[i]+" ");
11            System.out.println();
12    }//putArr
13    public static void main (String args []){
14        int A [ ] = {-5, 29, 0, -45, 55};
15        System.out.println ("Массив A:");
16        putArr(A);
17        System.out.println(minElArr(A));
18        int B [ ] = {3, -7, 5, 9, -10, 7, -3};
19        System.out.println ("Массив B:");
20        putArr(B);
21        System.out.println(minElArr(B));
22    }//main
23 }//class
```

saved

Рисунок 2.4 – Установка точки останова в программе

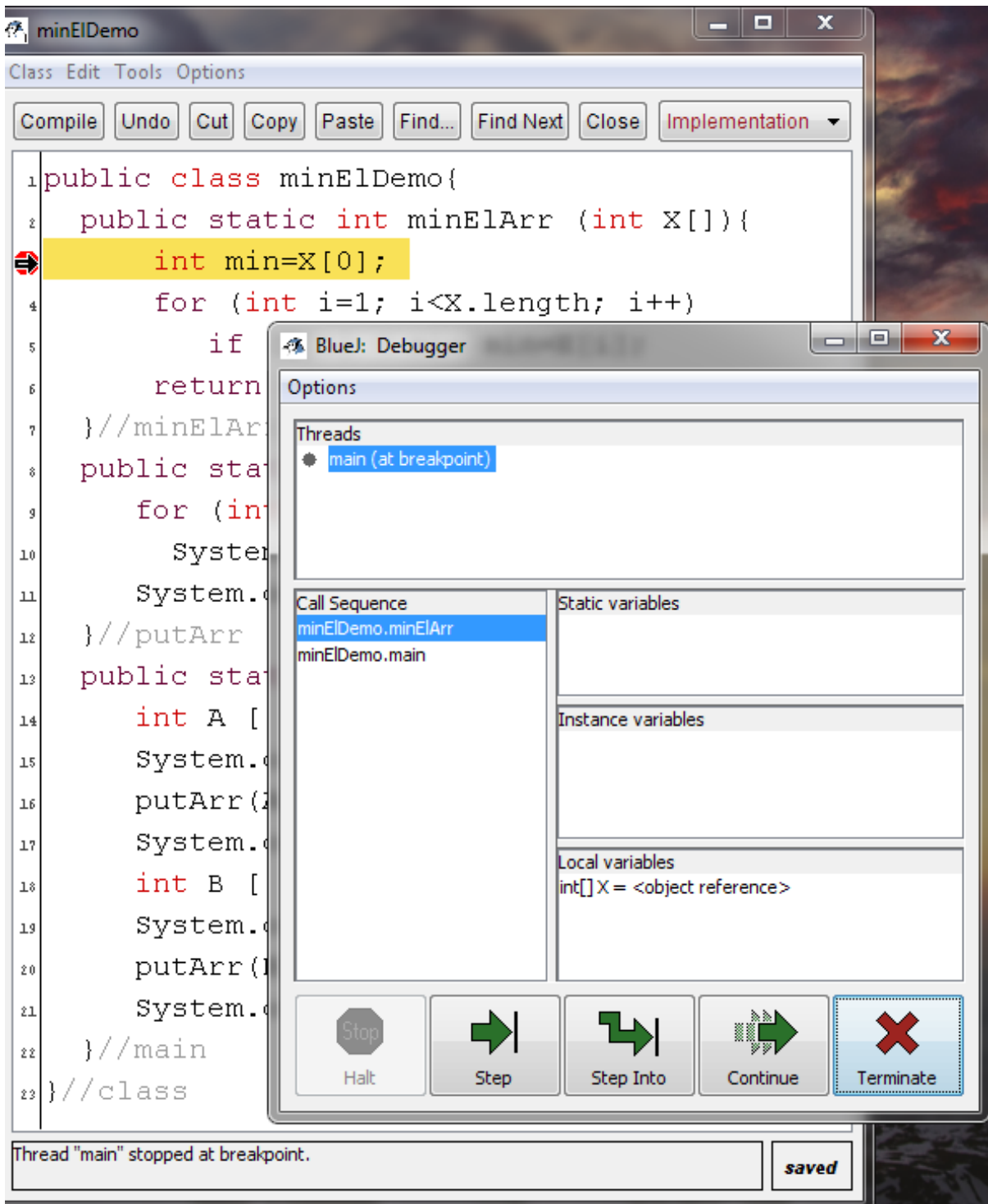


Рисунок 2.5 – Окно отладчика после запуска метода main()

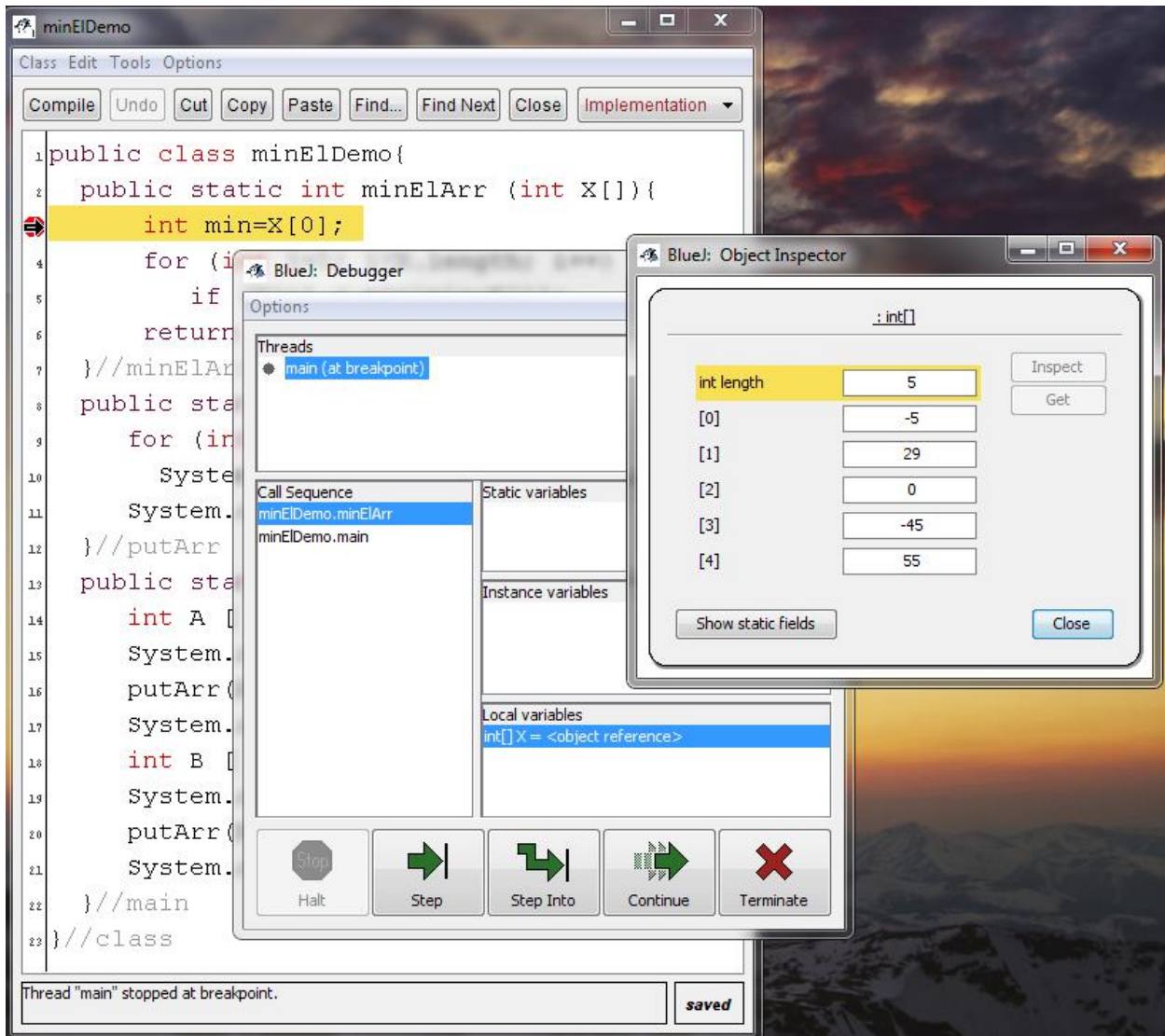


Рисунок 2.6 – Просмотр элементов массива

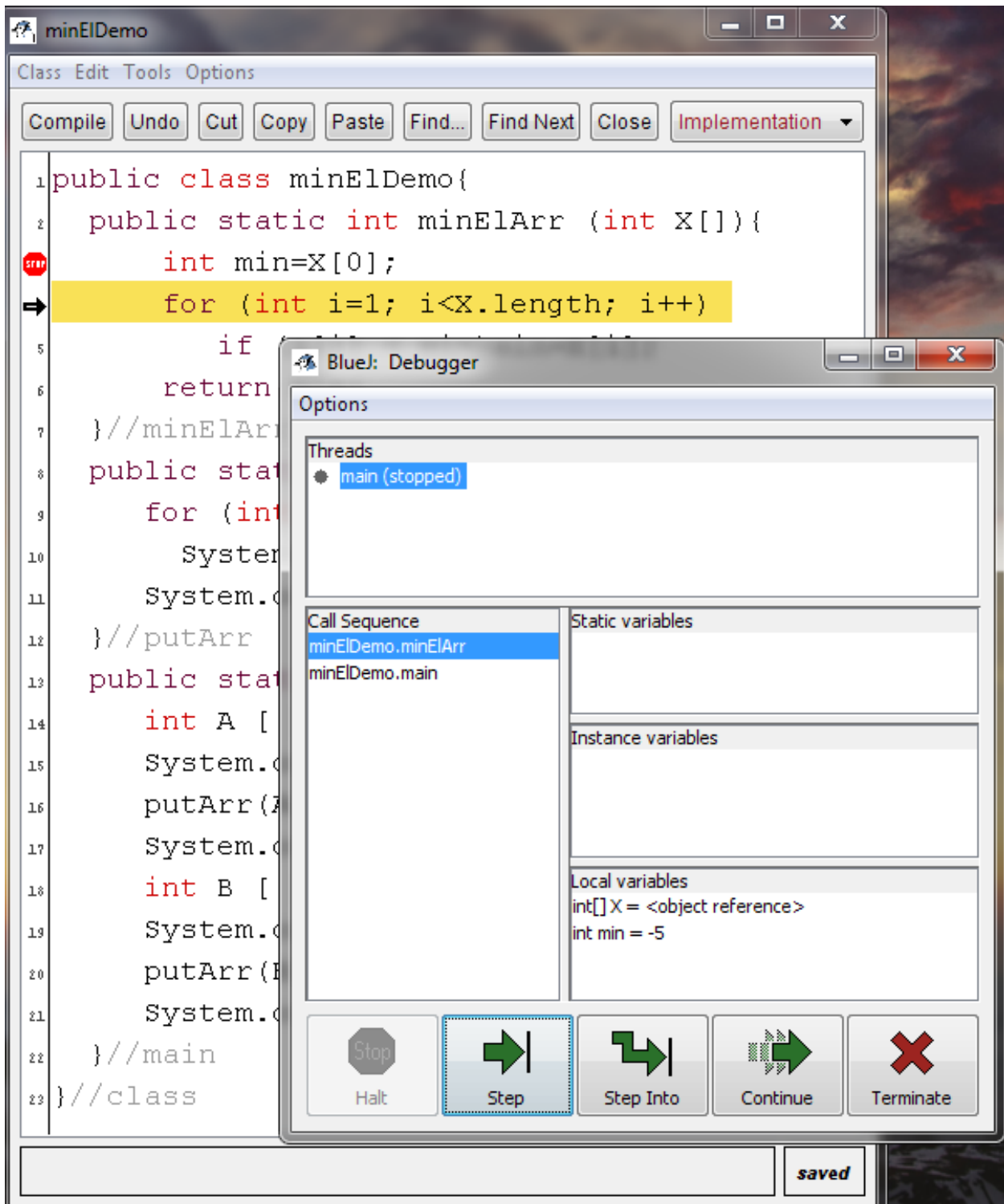


Рисунок 2.7 – Пошаговое выполнение (с точки останова) – шаг 1

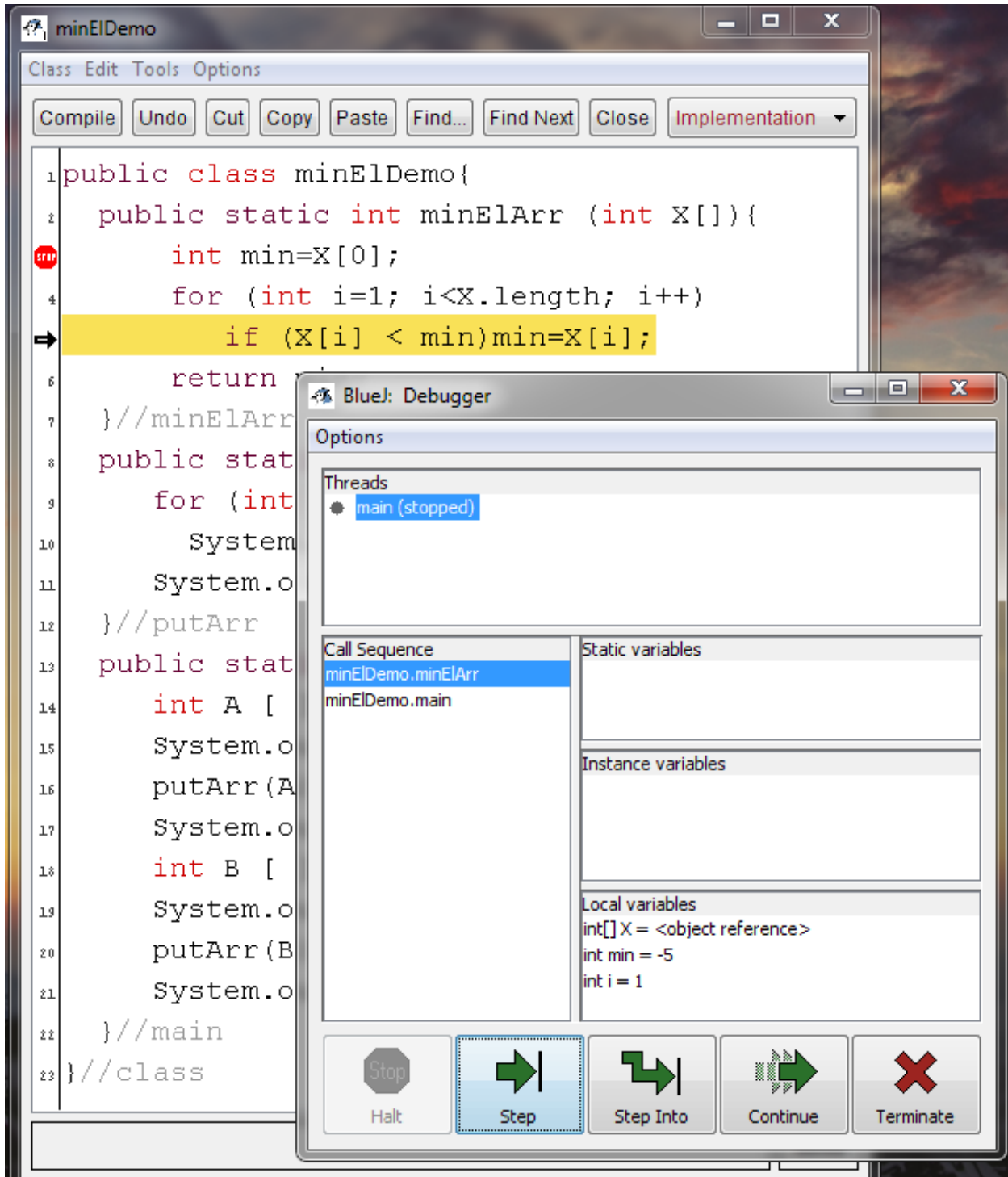


Рисунок 2.8 – Пошаговое выполнение – шаг 2

### 2.4.3. Подпрограммы

Подпрограмма – это отдельная функционально независимая часть программы.

Подпрограммы решают три важные задачи:

- 1) избавляют от необходимости многократно повторять в тексте программы аналогичные фрагменты;
- 2) улучшают структуру программы, облегчая ее понимание;
- 3) повышают устойчивость к ошибкам программирования и непредвиденным последствиям при модификациях программы.

Подпрограммы могут быть стандартными, т.е. определенными системой, и собственными, т.е. определенными программистом.

Стандартная подпрограмма (процедура или функция) - подпрограмма, включенная в библиотеку программ ЭВМ, доступ к которой обеспечивается средствами языка программирования. Вызывается она по имени с заданием фактических параметров соответствующих типов, заданных в описании данной процедуры в библиотеке процедур и функций.

Подпрограмма может иметь или не иметь параметры.

Формальные параметры подпрограммы указываются в описании подпрограммы с указанием соответствующих типов (после имени подпрограммы в круглых скобках).

Фактические параметры указываются при вызове подпрограммы (после имени подпрограммы в круглых скобках).

Компилятор проверяет соответствие числа фактических параметров числу формальных параметров, а, также, соответствие типов фактических параметров типам формальных параметров.

В Java в качестве фактических параметров в подпрограммы (методы) могут передаваться константы, переменные простых типов (передаются по значению), переменные сложных типов (передаются по ссылке) – объекты (в том числе, массивы).

Передача по значению: передается только значение переменной, указанной в качестве фактического параметра. Это значение присваивается внутренней (локальной) переменной метода, и все действия затем производятся с этой внутренней переменной, никак не отражаясь на значении переменной, использованной в качестве фактического параметра. Т.е. метод не может изменить значение фактического параметра, переданного по значению.

Передача по ссылке: передается адрес переменной, и все действия в методе производятся именно над переменной, имя которой указано в качестве фактического параметра. Т.е. метод может изменить значение фактического параметра, переданного по ссылке (адресу).

В других языках программирования, например Turbo Pascal, нет ограничения (там переменная любого типа передается по ссылке, если перед соответствующим формальным параметром в описании подпрограммы стоит слово var, а если этого слова нет – по значению).

Если подпрограмма возвращает значение, ее называют подпрограммой-функцией. Если подпрограмма просто выполняет последовательность действий, но не возвращает значения, то ее называют подпрограммой-процедурой.



В Java подпрограммы называются методами. Если метод возвращает значение, то в нем обязательно должен присутствовать оператор **return** **<выражение>;**, а в заголовке метода перед именем метода указывается тип возвращаемого значения. При этом тип **<выражения>** и тип возвращаемого значения должны совпадать.

Если метод не возвращает значения, в соответствующем месте заголовка метода указывается тип **void**.

Переменные, определенные внутри метода, являются локальными для этого метода и не доступны из других методов. Здесь действует следующее общее правило.

Переменные могут быть объявлены внутри любого блока. Блок – это последовательность операторов, заключенная в фигурные скобки. Как только блок кода создан, он становится логическим целым (несколько операторов превращаются в один составной оператор).

Блок определяет область видимости (действия) имен переменных и время их жизни.

Внутри блока переменные могут быть объявлены в любой точке, но область их действия начинается только после того, как они объявлены.

Переменные, объявленные в блоке, создаются вновь при каждом входе в блок и «разрушаются» при выходе из него, поэтому данные переменные недоступны (невидимы) коду, определенному вне этого блока.

Каждый раз, когда запускается новый блок, создается новая область видимости (действия).

Блоки и, соответственно, области действия могут быть вложенными.

Внешняя область действия включает внутреннюю. Это означает, что переменные, объявленные во внешней области, будут видимы и во внутренней.

Обратное не верно, т.е. из внешнего блока нет доступа к переменным, объявленным во внутреннем блоке.

## 2.5. Порядок выполнения работы

1) Составьте алгоритмы (схемы) программы и методов, к которым она обращается. В программе должно быть предусмотрено задание значений переменных, вывод в окно терминала результатов с пояснениями.

2) В соответствии с алгоритмом составьте программу на языке Java;

3) Проведите отладку программы. Используйте для этого возможности отладчика BlueJ (установка точек останова, пошаговое выполнение программы, визуализация значений переменных).

4) Проверьте работу программы при различных значениях параметров A B и n (не менее, чем на двух наборах значений), результаты приведите в отчете.

## 2.6. Варианты заданий

Варианты заданий представлены в таблице 2.1.

Номер варианта задания  $V$  необходимо вычислить по формуле

$$\text{int } V = (N \leq 14) ? \text{variant} : N \% 14;$$

где  $N$  – номер студента в списке группы,

$\text{variant}$  – номер варианта задания в таблице 2.1.

Таблица 2.1 – Варианты заданий

Но- мер вари- анта	Базовый тип массива	Функция (метод 1) должна возвращать значение:	Метод сортировки (для метода 2)	Вид сорти- ровки (для мето- да 2)	Процедура обработки массива (метод 3)
1	int	Сумма положительных элементов массива	Пузырек 1	По возр.	Увеличить на $A$ значения элементов с четными индексами
2	byte	Произведение отрицательных элементов массива	Пузырек 2	По убыв.	Обнулить элементы, значение которых находится в интервале $[A, B[$
3	short	Минимальный среди положительных элементов	Прямой выбор	По возр.	Заменить элементы с нечетным значением остатком от их деления на 2
4	long	Максимальный среди отрицательных элементов	Прямые вставки	По убыв.	Увеличить на $B$ значения всех элементов, меньших $A$ .
5	double	Максимальный по модулю элемент	Прямой выбор	По возр.	Заменить все элементы, значение которых находится в интервале $[A, B]$ значением максимального по модулю элемента.
6	float	Количество отрицательных элементов	Пузырек 1	По убыв.	Уменьшить все отрицательные элементы на $A$ , если количество отрицательных элементов $\leq B$
7	char	Количество элементов, равных $A$ .	Пузырек 2	По возр.	Заменить на $B$ все элементы, значение которых равно $A$ .

## Продолжение таблицы 2.1

Но- мер вари- анта	Базовый тип массива	Функция (метод 1) должна возвращать значение:	Метод сортировки (для метода 2)	Вид сорти- ровки (для мето- да 2)	Процедура массива (метод 3) обработки
8	int	Сумма первых $m$ элементов массива	Прямой выбор	По убыв.	Заменить положительные элементы, значение которых больше $A$ их квадратными корнями.
9	float	Номер минимального элемента	Пузырек2	По убыв.	Заменить все положительные элементы массива на $0$ , отрицательные – на $1$ .
10	byte	Номер максимального элемента	Прямые вставки	По возр.	Переставить местами максимальный элемент и элемент массива со значением $B$ (если таких элементов несколько, то первый из них)
11	short	Сумма модулей элементов массива	Прямой выбор	По убыв.	Разделить на $A$ все отрицательные элементы массива, модуль которых больше $B$
12	char	Количество элементов, значение которых находится в интервале $[A,B]$	Пузырек 1	По возр.	Поменять местами первый слева элемент со значением $A$ и первый справа элемент со значением $B$ .
13	double	Произведение элементов с четными индексами	Пузырек 2	По убыв.	Изменить знак каждого элемента, модуль которого находится в интервале $[A,B]$
14	int	Сумма последних $m$ элементов.	Прямые вставки	По возр.	Для первых $A$ элементов массива заменить все элементы с четным значением нулями, а все элементы с нечетным значением единицами

## 2.7. Контрольные вопросы

- 1) Что представляет собой одномерный массив? Каковы основные характеристики массива?
- 2) Как осуществляется доступ к элементам массива?
- 3) Что такое индекс массива? Какие допустимы типы индексов массива?
- 4) В массиве 100 элементов. Укажите номера первого и последнего элементов.
- 5) Как выполнить создание массива с помощью инициализации?
- 6) Опишите действия (нарисуйте изменения в ОП), которые выполняет Java-машина при исполнении каждого из следующих операторов:

```
int [] X;
X={1,-2,3};
int Y[] = new Y[5];
for (int i=0; i<Y.length; i++) Y[i]=i+1;
int [] D, C;
C=X;
D=Y;
int E[]=C;
```

- 7) Какие значения вернут операции сравнения X==C; X==D; Y==D; Y==E?
- 8) Сформулируйте алгоритм нахождения минимального и максимального элементов в числовом массиве.
- 9) Что собой представляет процедура? Когда целесообразно использовать процедуры? Как задать процедуру в Java?
- 10) Что собой представляет функция? Когда целесообразно использовать функции? Как задать функцию в Java?
- 11) Что такое формальные параметры подпрограммы (метода)? Что такое фактические параметры подпрограммы (метода)? Какие соответствия между формальными и фактическими параметрами проверяет компилятор?
- 12) Что такое параметры-значения и параметры-переменные? Чем они отличаются друг от друга? Как в Java осуществляется передача фактических параметров в методы.
- 13) Можно ли в методе изменить значение элемента массива, переданного в метод, как параметр?
- 14) Можно ли в методе изменить значение фактического параметра простого типа?
- 15) Если метод возвращает значение, какой оператор обязательно должен присутствовать в нем?
- 16) Если метод не возвращает значение, а просто выполняет какие-то действия, как это отражается в его заголовке и теле?
- 17) Что такое локальная переменная метода? Приведите пример из своей программы. Что вы можете сказать об области действия и «времени жизни» такой переменной? Какое общее правило Java здесь действует?

### **3. ЛАБОРАТОРНАЯ РАБОТА № 9. ИССЛЕДОВАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ ОПЕРАЦИИ НАД МАТРИЦАМИ**

#### **3.1. Цель работы**

Целью данной работы является исследование циклических алгоритмов и программ, осуществляющих типичные операции над двумерными массивами (матрицами), получение навыков разработки и использования подпрограмм (методов-процедур и методов-функций).

#### **3.2. Постановка задачи**

Разработать программу, реализующую операции над матрицами по заданию, указанному в таблице 3.1, в соответствии с номером варианта. Программа должна удовлетворять нижеперечисленным требованиям.

В основной программе (методе main) нужно инициализировать две квадратные матрицы (число строк равно числу столбцов), содержащих элементы заданного числового типа.

Основная программа (метод main) должна вызывать три вспомогательные подпрограммы (три метода).

Первый метод-функция предназначен для вычисления скалярного значения в соответствии с заданием, указанным в столбце 3 таблицы вариантов. Под скалярным значением понимается значение простого типа.

Второй метод-функция предназначен для вычисления векторного значения в соответствии с заданием, указанным в столбце 4 таблицы вариантов. Под векторным значением понимается значение типа «одномерный массив».

Третий метод-функция должен возвращать матрицу, являющуюся результатом выполнения операции над матрицами, заданной в столбце 5 таблицы вариантов.

Кроме того, рекомендуется включить в проект методы-процедуры для вывода на экран матрицы и одномерного массива.

Основная программа (метод main) должна выполнять следующие действия:

1) инициализировать две квадратные матрицы A и B;

2) вывести:

исходные матрицы A и B;

результат выполнения метода-функции 1 для матрицы A и матрицы B;

результат выполнения метода-функции 2 для матрицы A и матрицы B;

результат выполнения метода-функции 3 для матрицы A и матрицы B.

#### **3.3. Внеаудиторная подготовка**

Для подготовки к лабораторной работе следует изучить краткие теоретические сведения, приведенные в пункте 3.4 методических указаний. Рекомендуется также ознакомиться с [1] (с.72-79).

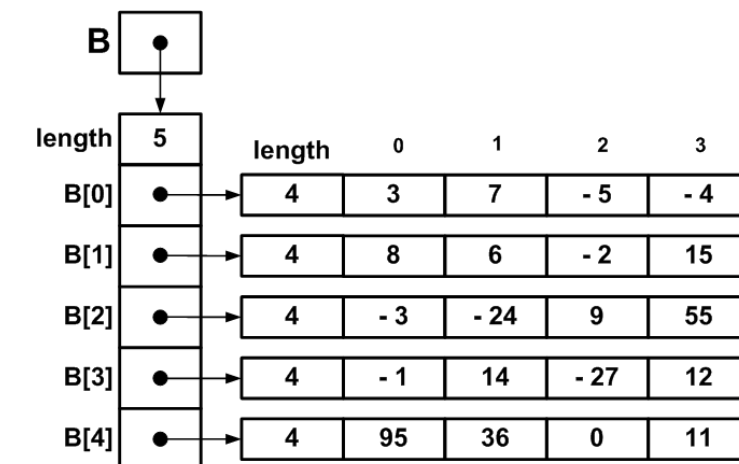
### 3.4. Краткие теоретические сведения

#### 3.4.1. Многомерные массивы

Математический вектор может быть представлен одномерным массивом, а матрица – двумерным массивом.

Язык Java допускает массивы массивов – многомерные массивы [3].

На рисунке 3.1 изображен прямоугольный двумерный массив – матрица – с именем В (в общем случае, двумерные массивы в Java не обязательно должны содержать одинаковое число элементов в строке).



B[0][0]=3  
B[2][1]= - 24

*B.length=5 – число строк матрицы*

*B[0].length=B[1].length=B[2].length==B[3].length=  
=B[4].length=4 – число столбцов матрицы*

Рисунок 3.1 – Двумерный массив

В переменной В содержится адрес одномерного массива, *i*-ый элемент которого (*i*=0...4), в свою очередь, содержит адрес одномерного массива элементов *i*-ой строки. Другими словами, В ссылается на одномерный массив, каждый элемент которого ссылается на одномерный массив элементов соответствующей строки матрицы (адрес и ссылка – синонимы).

Тип многомерного массива задается указанием двух и более пар квадратных скобок. Так, для двумерного массива нужны две пары квадратных скобок.

Объявление двумерного массива:

**type var\_name [ ] [ ];**

или

**type [ ] [ ] var\_name;**

type – базовый тип массива (тип элементов массива),

var\_name – имя переменной массива.

Двумерный массив, как и одномерный, можно инициализировать:

```
int B [ ] [ ] = {
    {3, 7, - 5, - 4},
    {8, 6, - 2, 15},
    {- 3, - 24, 9, 55},
    {- 1, 14, - 27, 12},
    {95, 36, 0, 11}
};
```

Объявлен и создан двумерный массив, в котором пять строк и четыре столбца. Элементы получили начальные значения согласно списку констант.

Другой способ создания двумерного массива при помощи – процедуры new:

```
int [ ] [ ] B = new int[2] [3];
```

Начальные значения элементов задаются по умолчанию.

Для перебора элементов одномерного массива (вектора) мы использовали цикл for. Для перебора элементов двумерного массива (матрицы) требуется конструкция, состоящая из двух циклов for (основного и вложенного). В основном цикле будет изменяться номер строки (первый индекс элемента матрицы), а во вложенном – номер столбца (второй индекс элемента матрицы), или наоборот.

Пример 1. Ниже приведен текст метода, предназначенного для вывода элементов матрицы:

```
public static void putMatr (int Y[ ][ ]){
    for (int i=0; i<Y.length; i++) {
        for (int j=0; j<Y[0].length; j++)
            System.out.printf ("% 5d",Y[ i ][ j ]);
        System.out.println();
    } //for i
} //putMatr
```

Пример 2. Разработать программу обработки матриц при помощи метода, возвращающего массив X, содержащий максимальные элементы столбцов матрицы Y. Алгоритм метода изображен на рисунке 3.2.

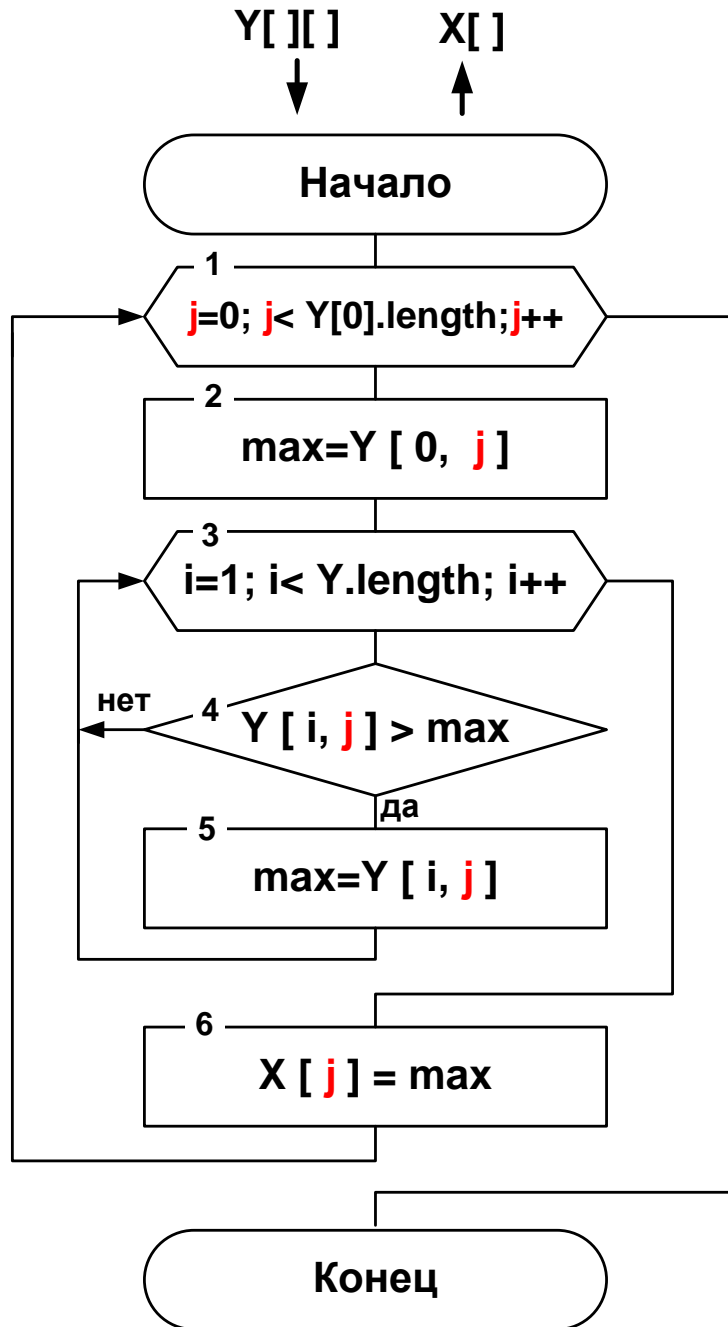


Рисунок 3.2 – Схема метода (функции) обработки массива

Текст программы:

```

public class MatrArrDemo{
    public static int[] MaxInColomn (int Y[][]){
        int X[]=new int[Y[0].length]; // создает массив X,
        //состоящий из Y[0].length элементов (слов памяти)
        for (int j=0; j<Y[0].length; j++) {
            int max=Y[0][j];
            for (int i=1; i<Y.length; i++)
                if (Y[i][j]>max) max=Y[i][j];
        }
    }
}
  
```



```

        X[j]=max;
    } //for j
    return X; // метод может вернуть переменную любого типа
} //MaxInColomn
public static void putMatr (int Y[][]){
    for (int i=0; i<Y.length; i++) {
        for (int j=0; j<Y[0].length; j++)
            System.out.printf ("% 5d",Y[i][j]);
        System.out.println();
    } //for i
} //putMatr
public static void putArr (int X[]){
    for (int i=0; i<X.length; i++)
        System.out.printf ("% 5d",X[i]);
    System.out.println();
} //putArr
public static void main (String args []){
    int Y [ ] [ ] = {
        {3, 72, -5, -4},
        {8, 6, -2, -15},
        {-3, -24, 9, -55},
        {-1, 14, -27, -1},
        {95, 36, 0,-11}
    };
    int A [ ] [ ] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12}
    };
    System.out.println("Матрица Y:");
    putMatr(Y);
    int X[]= MaxInColomn(Y);
    System.out.println("Массив X для матрицы Y:");
    putArr(X);
    System.out.println("Матрица A:");
    putMatr(A);
    System.out.println("Массив X для матрицы A:");
    putArr(MaxInColomn(A));
} //main
} //class

```

Большое количество задач связано с обработкой части элементов квадратной матрицы (количество строк совпадает с количеством столбцов), например, рассматриваются только элементы главной диагонали, над дополнительной диагональю (соединяющей элементы  $A_{n,1}$  и  $A_{1,n}$  и т.д). При решении таких задач возникает проблема, определения границ изменения индексов обрабатываемых элементов. Ниже приведем ряд типичных случаев с указанием соответствующих начальных и конечных значений в заголовках циклов (рисунки 3.3):

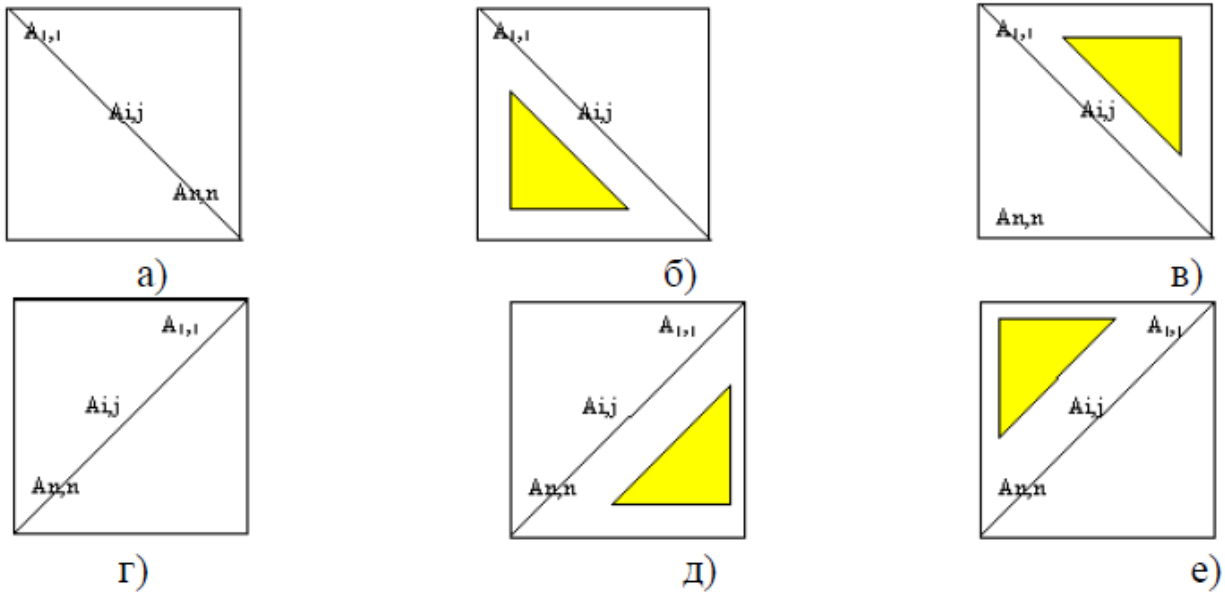


Рисунок 3.3 – Определение индексов элементов квадратной матрицы

- а) элементы, принадлежащие главной диагонали:  
 // обнуление элементов главной диагонали  
 for (int i=0; i<A.length; i++) A[i][i]=0;
- б) элементы под главной диагональю:  
 // обнуление элементов под главной диагональю  
 for (int i=1; i<A.length; i++)  
 for (int j=0; j<i; j++) A[i][j]=0;
- в) элементы над главной диагональю:  
 // обнуление элементов над главной диагональю  
 for (int i=0; i<A.length-1; i++)  
 for (int j=i+1; j< A.length; j++) A[i][j]=0;
- г) элементы второй диагонали:  
 // обнуление элементов второй диагонали  
 for (int i=0; i<A.length; i++) A[i][ A.length-i-1]=0;
- д) элементы под второй диагональю:  
 // обнуление элементов под второй диагональю  
 for (int i=1; i<A.length; i++)  
 for (int j=A.length-i; j<A.length; j++) A[i][j]=0;
- е) элементы над второй диагональю:  
 // обнуление элементов над второй диагональю  
 for (int i=0; i<A.length; i++)  
 for (int j=0; j<A.length-i-1; j++) A[i][j]=0;

### 3.4.2. Операции над матрицами

Произведением матрицы на число называется матрица, полученная из исходной матрицы умножением каждого ее элемента на заданное число.

Суммой матриц  $A$  и  $B$  одного и того же размера называется матрица  $C=A+B$  такого же размера, получаемая из исходных матриц путем сложения соответствующих элементов.

Произведением матрицы  $A_{m \times n}$  на матрицу  $B_{n \times k}$  называется матрица  $C_{m \times k}$ , такая, что элемент матрицы  $C_{ij}$  равен сумме произведений элементов  $i$ -ой строки матрицы  $A$  на соответствующие элементы  $j$ -го столбца матрицы  $B$ .

Транспонирование матрицы – это операция над матрицей, когда ее строки становятся столбцами с теми же номерами.

### 3.5. Порядок выполнения работы

1) Составьте алгоритмы (схемы) программы и методов, к которым она обращается. В программе должно быть предусмотрено задание значений переменных, вывод в окно терминала результатов с пояснениями.

2) В соответствии с алгоритмом составьте программу на языке Java (в методе, который выводит в окно терминала матрицу, используйте оператор форматной печати `printf` для вывода элементов матрицы);

3) Проведите отладку программы. Используйте для этого возможности отладчика BlueJ (установка точек останова, пошаговое выполнение программы, визуализация значений переменных).

4) Проверьте работу программы при различных значениях параметров  $A$ ,  $B$  и  $n$  (не менее, чем на двух наборах значений), результаты приведите в отчете.

### 3.6. Варианты заданий

Варианты заданий представлены в таблице 3.1.

Номер варианта задания  $V$  необходимо вычислить по формуле

$$\text{int } V = (N \leq 14) ? \text{variant} : N \% 14;$$

где  $N$  – номер студента в списке группы,

`variant` – номер варианта задания в таблице 3.1.

Таблица 3.1 – Варианты заданий

Номер варианта	Базовый тип матриц (тип элементов )	Функция 1 (метод 1) с параметром А	Функция 2 (метод 2) с параметром А	Функция 3 (метод 3) с параметрами А, В, m
1	byte	Сумма элементов, расположенных под главной диагональю	Вектор, содержащий минимальные значения каждой строки матрицы А	Матрица: $m(A+B)$
2	short	Произведение элементов, расположенных под главной диагональю	Вектор, содержащий число нулевых элементов для каждой строки матрицы	Матрица: $A*B$
3	int	Минимальный из элементов, расположенных над главной диагональю	Вектор, содержащий число нулевых элементов для каждого столбца матрицы	Матрица: $A^t$
4	long	Максимальный из элементов, расположенных под главной диагональю	Вектор, содержащий значения сумм элементов каждой строки матрицы	Матрица: $(A+B)/m$
5	float	Сумма элементов, расположенных под второй (не главной) диагональю	Вектор, содержащий значения произведений элементов каждой строки матрицы	Матрица: $A*B$
6	double	Произведение элементов, расположенных под второй диагональю	Вектор, содержащий значения сумм элементов каждого столбца матрицы	Матрица: $B^t$
7	byte	Минимальный из элементов, расположенных над второй диагональю	Вектор, содержащий значения произведений элементов каждого столбца матрицы	Матрица: $m(A+B)$

Продолжение таблицы 3.1

Номер варианта	Базовый тип матриц (тип элементов)	Функция 1 (метод 1) с параметром А	Функция 2 (метод 2) с параметром А	Функция 3 (метод 3) с параметрами А, В, m
8	short	Максимальный из элементов, расположенных под второй диагональю	Вектор, содержащий число положительных элементов для каждой строки матрицы	Матрица: $A * B$
9	int	Максимальный из положительных ( $>0$ ) элементов главной диагонали	Вектор, содержащий число отрицательных элементов для каждого столбца матрицы	Матрица: $A^t$
10	long	Минимальный из отрицательных ( $<0$ ) элементов второй диагонали	Вектор, содержащий максимальные по модулю элементы каждой строки матрицы	Матрица: $(A+B)/m$
11	float	Первый отрицательный элемент среди элементов, расположенных над главной диагональю	Вектор, содержащий максимальные по модулю элементы каждого столбца матрицы	Матрица: $A * B$
12	double	Последний положительный элемент среди элементов под главной диагональю	Вектор, $i$ -ый элемент которого равен 1, если в $i$ -ой строке матрицы есть отрицательные элементы, 0 в противном случае.	Матрица: $B^t$
13	float	Число положительных элементов под главной диагональю	Вектор, $i$ -ый элемент которого равен 1, если в $i$ -ом столбце матрицы есть неотрицательные элементы, 0 в противном случае.	Матрица: $m(A+B)$

Продолжение таблицы 3.1

Номер варианта	Базовый тип матриц (тип элементов)	Функция 1 (метод 1) с параметром А	Функция 2 (метод 2) с параметром А	Функция 3 (метод 3) с параметрами А, В, m
14	double	Число отрицательных элементов над главной диагональю	Вектор, содержащий средние арифметические значения для каждого столбца матрицы	Матрица: $A \cdot B$

### 3.7. Контрольные вопросы

1) Какой тип данных подходит для представления матриц? Опишите структуру этого типа в Java.

2) Назовите способы создания многомерных (в частности, двумерных) массивов в Java.

3) Как осуществляется доступ к элементам матрицы? Какая операторная структура используется для того, чтобы получить доступ последовательно ко всем элементам матрицы?

4) Задана матрица:

```
int Y [][] = {
    {3, 72, -5, -4},
    {8, 6, -2, -15},
    {-3, -24, 9, -55},
    {-1, 14, -27, -1},
    {95, 36, 0, -11}
};
```

Как в программе определить число строк и число столбцов в матрице А?

5) Дайте определение операций умножения матрицы на константу и сложения двух матриц. Что вы можете сказать по поводу размерности суммируемых матриц? Приведите пример фрагмента программы, в котором осуществлялось бы сложение двух матриц и умножение результирующей матрицы на константу.

6) Дайте определение операции транспонирования матрицы. Приведите пример фрагмента программы, в котором осуществлялось бы транспонирование квадратной матрицы.

7) Дайте определение операции умножения матриц и приведите соответствующую формулу. Как должны быть согласованы размерности матриц? Приведите пример фрагмента программы, в котором осуществлялось бы умножение двух матриц.

8) Как выделить элементы, находящиеся в следующих областях матрицы: на главной диагонали, под главной диагональю, над главной диагональю, на второй диагонали, под второй диагональю, над второй диагональю?

9) Что такое формальные и фактические параметры метода? Где они задаются? Какие способы передачи параметров вам известны?

10) Если матрица является параметром метода, то что на самом деле передается в метод? Какое общее правило Java здесь действует?

11) Что указывается в заголовке метода и что указывается в вызове метода? Приведите пример соответствия формальных и фактических параметров метода (из составленной вами программы). Какие проверки осуществляет здесь компилятор?

12) Можно ли изменить в методе значение элемента матрицы, переданной в метод в качестве фактического параметра?

13) Можно ли изменить в методе значение переменной простого типа, переданной в метод в качестве фактического параметра?

14) Значения каких типов может возвращать метод в Java? Где задается тип возвращаемого значения?

15) Если метод возвращает значение, какой оператор обязательно должен присутствовать в нем?

16) Если метод не возвращает значение, а просто выполняет какие-то действия, как это отражается в его заголовке и теле?

17) Что такое локальная переменная метода? Приведите пример из своей программы. Что вы можете сказать об области действия и «времени жизни» такой переменной? Какое общее правило Java здесь действует?

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ноутон П. Java 2/ П.Ноутон, Г.Шилдт. – СПб.: БХВ-Петербург, 2007. – 1072 с.
2. Основные виды сортировок и примеры их реализации. Академия Яндекса. – Электронный ресурс. – Режим доступа: <https://academy.yandex.ru/posts/osnovnye-vidy-sortirovok-i-primery-ikh-realizatsii> .
3. Глушаков С. В Программирование на Java 2 : учеб. пособие для студ. вузов/ С. В. Глушаков. – Харьков: Фолио, 2003. – 544 с.

