Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Севастопольский государственный университет»

Методические указания
к лабораторным работам по дисциплине
«Алгоритмизация и программирование»
для студентов дневной и заочной форм обучения направлений
09.03.02 – «Информационные системы и технологии»
и 09.03.03 – «Прикладная информатика»

часть 3

Севастополь

2021

УДК 004.42 (075.8)

Методические указания к лабораторным работам по дисциплине «Алгоритмизация и программирование» для студентов дневной и заочной форм обучения направлений 09.03.02 – «Информационные системы и технологии» и 09.03.03 – «Прикладная информатика», часть 3 / Сост. В. Н. Бондарев, Т. И. Сметанина, А.Ю. Абрамович – Севастополь: Изд-во СевГУ, 2021. – 30 с.

Методические указания предназначены для проведения лабораторных работ и обеспечения самостоятельной подготовки студентов по дисциплине «Алгоритмизация и программирование». Целью методических указаний является обучение студентов основным принципам структурного программирования, навыкам разработки программ на языках Java и C/C++.

Методические указания составлены в соответствии с требованиями программы дисциплины «Алгоритмизация и программирование» для студентов направлений 09.03.02 – «Информационные системы и технологии» и 09.03.03 – «Прикладная информатика « и утверждены на заседании кафедры «Информационные системы» протокол № ____

Допущено научно-методической комиссией института информационных технологий и систем управления в технических системах в качестве методических указаний.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цели и задачи лабораторных работ

Основная цель выполнения настоящих лабораторных работ – получение практических навыков создания и отладки программ на языке Java. В результате выполнения лабораторных работ студенты должны углубить знания основных теоретических положений дисциплины «Алгоритмизация и программирование», решая практические задачи на ЭВМ.

Студенты должны получить практические навыки работы в интегрированной системе программирования Eclipse JDT, навыки разработки программ на языке Java.

Выбор вариантов и график выполнения лабораторных работ

Варианты заданий приведены в каждой лабораторной работе и уточняются преподавателем.

Лабораторная работа выполняется в два этапа. На первом этапе – этапе самостоятельной подготовки – студент должен выполнить следующее:

– проработать по конспекту и рекомендованной литературе, приведенной в конце настоящих методических указаний, основные теоретические положения лабораторной работы и подготовить ответы на контрольные вопросы;

– разработать алгоритм решения задачи и составить схему алгоритма;

– описать схему алгоритма;

– написать программу на языке Java и описать ее;

– оформить результаты первого этапа в виде заготовки отчета по лабораторной работе.

На втором этапе, выполняемом в лабораториях кафедры, студент должен:

– отладить программу;

– разработать и выполнить тестовые примеры.

Выполнение лабораторной работы завершается защитой отчета. Студенты должны выполнять и защищать работы **строго по графику**.

График уточняется преподавателем, ведущим лабораторные занятия.

Требования к оформлению отчета

Отчёты по лабораторной работе оформляются каждым студентом индивидуально. Отчёт должен включать: название и номер лабораторной работы; цель работы; вариант задания и постановку задачи; результаты выполнения программы; выводы по работе; приложения. Содержание отчета указано в методических указаниях к каждой лабораторной работе.

1 ЛАБОРАТОРНАЯ РАБОТА №1 «ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ И РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ НА JAVA»

1.1 Цель работы

Изучение структуры Java-программы.

Формирование навыков программирования алгоритмов линейной и разветвляющейся структуры на языке Java.

Исследование особенностей ввода-вывода значений стандартных типов на языке Java.

1.2 Краткие теоретические сведения

1.2.1 Запуск Eclipse, создание проекта и класса Java

Для запуска `Eclipse` скачайте архив по ссылке https://disk.sevsu.ru/index.php/s/S8eYRnrTDnxST7R, раскройте архив на свой рабочий компьютер (например, в корень диска `C`). Запустите **eclipse.exe.**

После первого запуска Eclipse отобразится диалоговое окно выбора рабочего пространства (workspace). Это каталог, в который сохраняются проекты пользователя. Рекомендуется каждому студенту создать свое рабочее пространство. В нашем случае – **C:\eclipse_202203\eclipse-workspace** (рисунок 1.1).

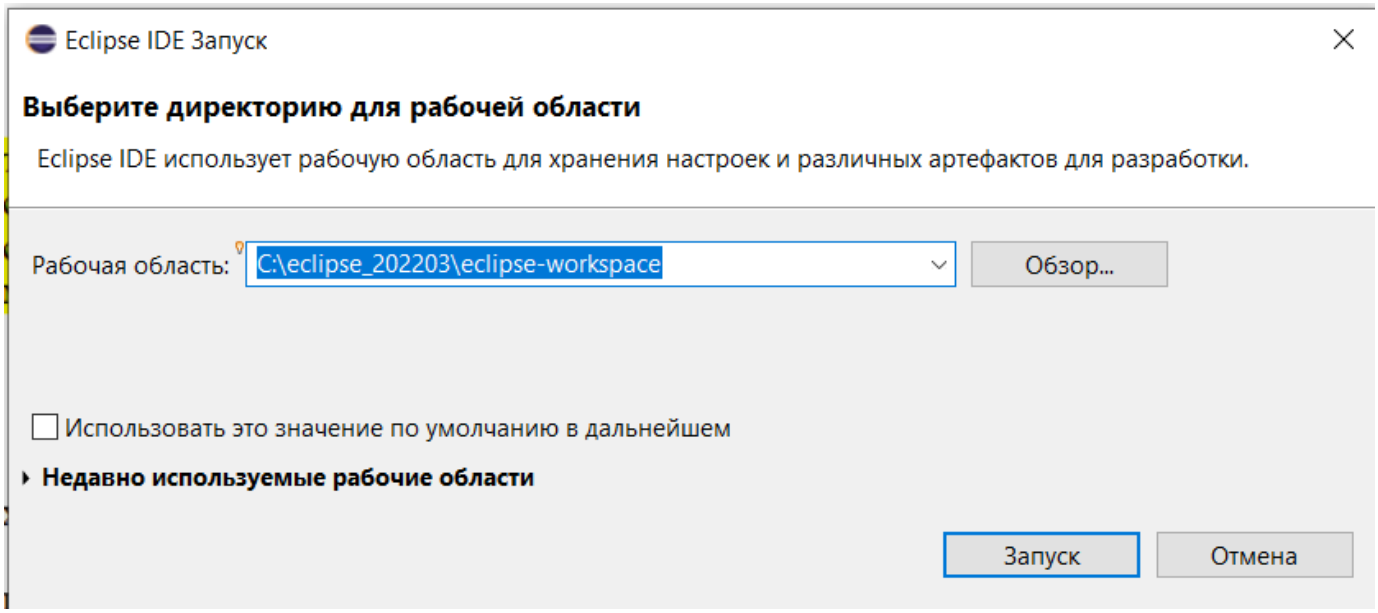

Рисунок 1.1 – Диалоговое окно выбора рабочего пространства

В окне приложения (рисунок 1.2) для создания нового проекта выберите в меню: **Файл > Новый > Проект Java** (рисунок 1.3).

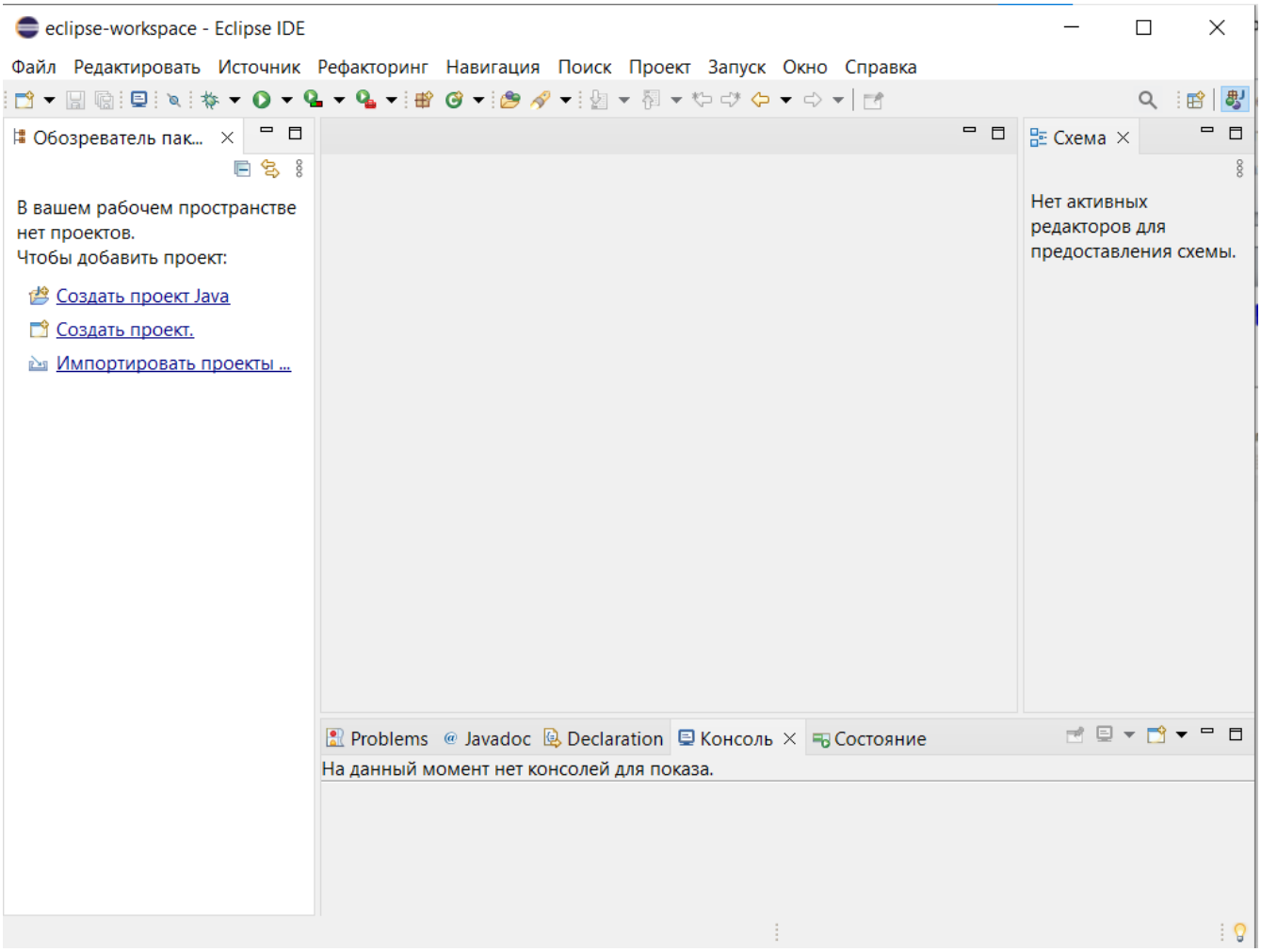

Рисунок 1.2 – Окно приложения

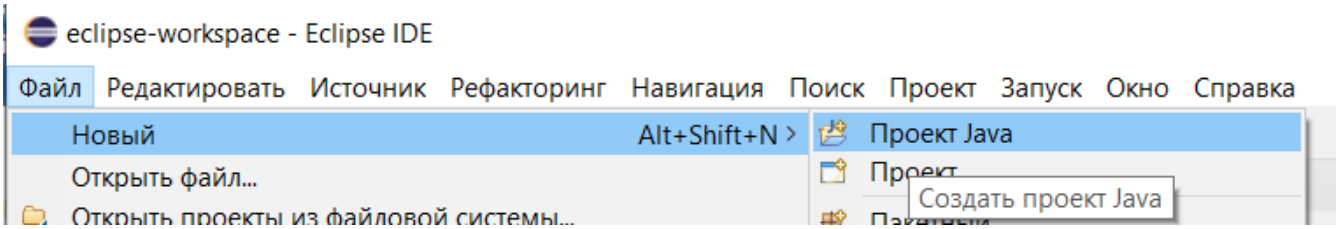

Рисунок 1.3 – Создание нового проекта – последовательность команд

Введите HelloWorld в поле «имя проекта» и нажмите «Готово» (рисунок 1.4). Вместо «HelloWorld» можно ввести любое имя, но **имя проекта и имя класса должны совпадать**.

Для создания класса Java в обозревателе пакетов на имени проекта нажмите правой кнопкой мыши (ПКМ) и выберите**: Создать>Класс** (рисунок 1.5).

В окне «Класс Java» введите в поле «имя» HelloWorld (или ваше имя) и выделите опцию «общедоступная статическая пустота(String[] args)» и нажмите «Готово» (рисунок 1.6).

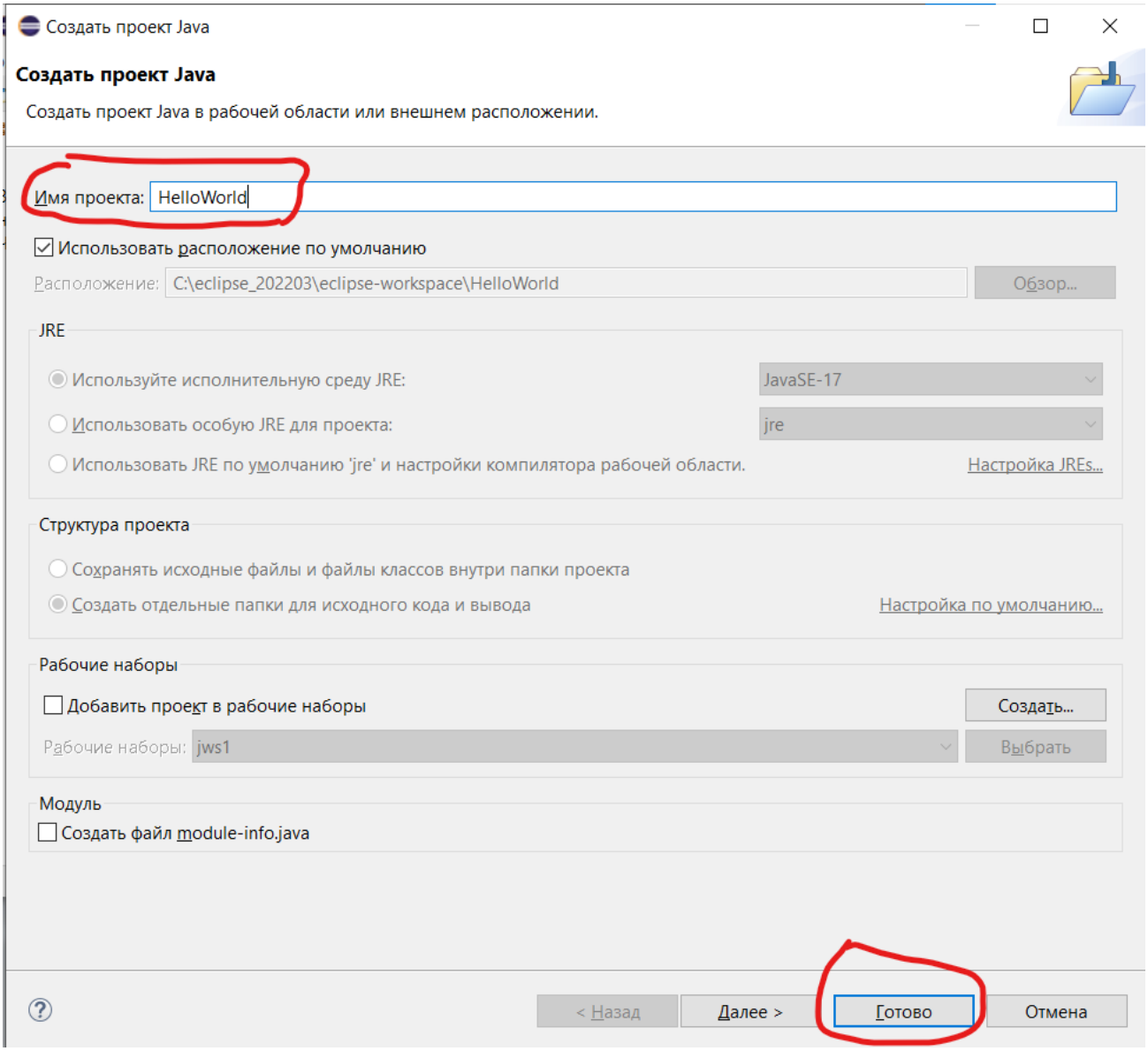

Рисунок 1.4 – Заполнение полей при создании нового проекта

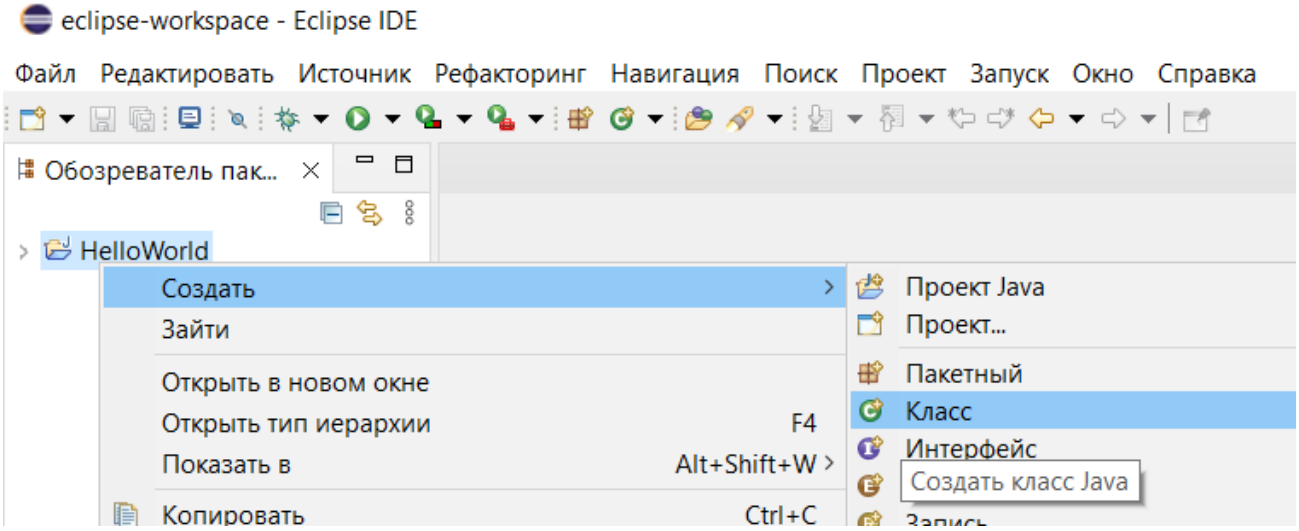

Рисунок 1.5 – Создание класса – последовательность команд

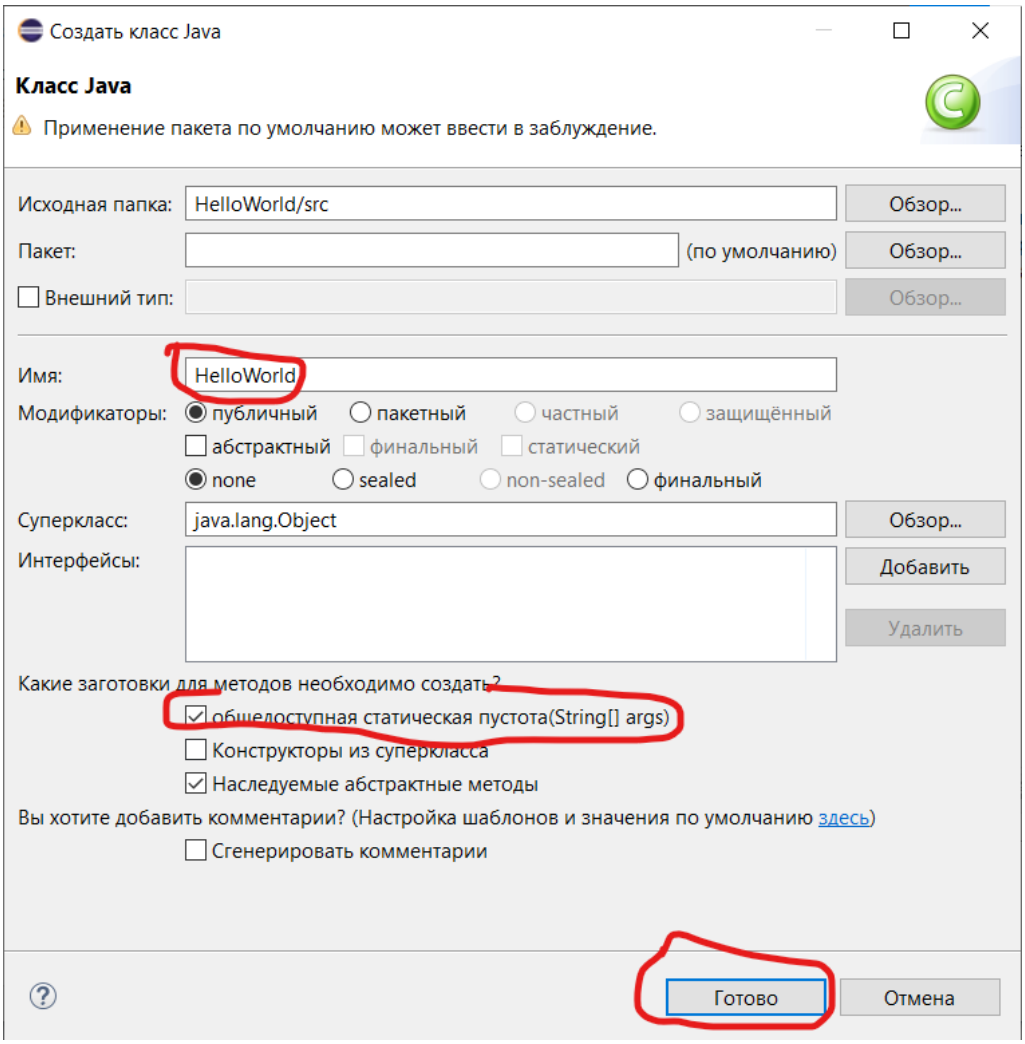

Рисунок 1.6 – Создание класса HelloWorld

После этого в проект будет добавлен файл HelloWorld.java. В методе main() добавьте следующий оператор: **`System.out.println(«Hello Eclipse!»);`** (рисунок 1.7).

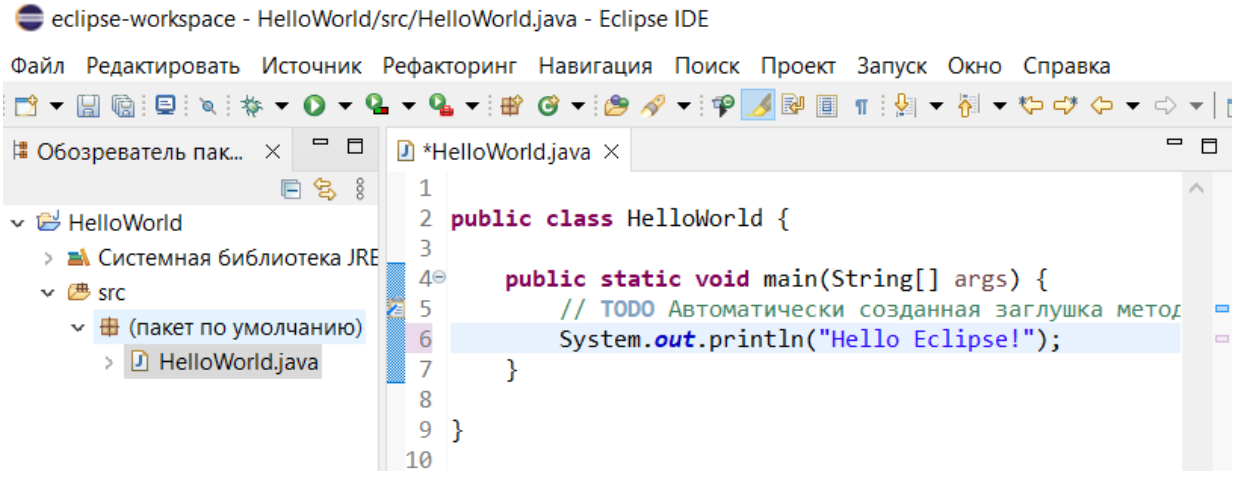

Рисунок 1.7 – Создание первой программы

Сохраните файл **Файл>Сохранить** (Ctrl-S) (рисунок 1.8).

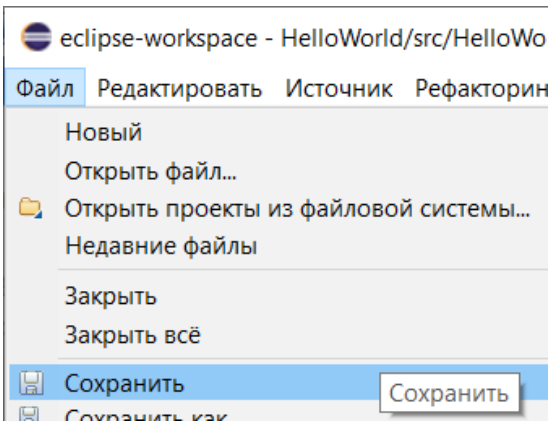

Рисунок 1.8 – Сохранение

Для запуска проекта нажмите правой кнопкой мыши на имени проекта **Выполнить как > Приложение Java** (рисунок 1.9).

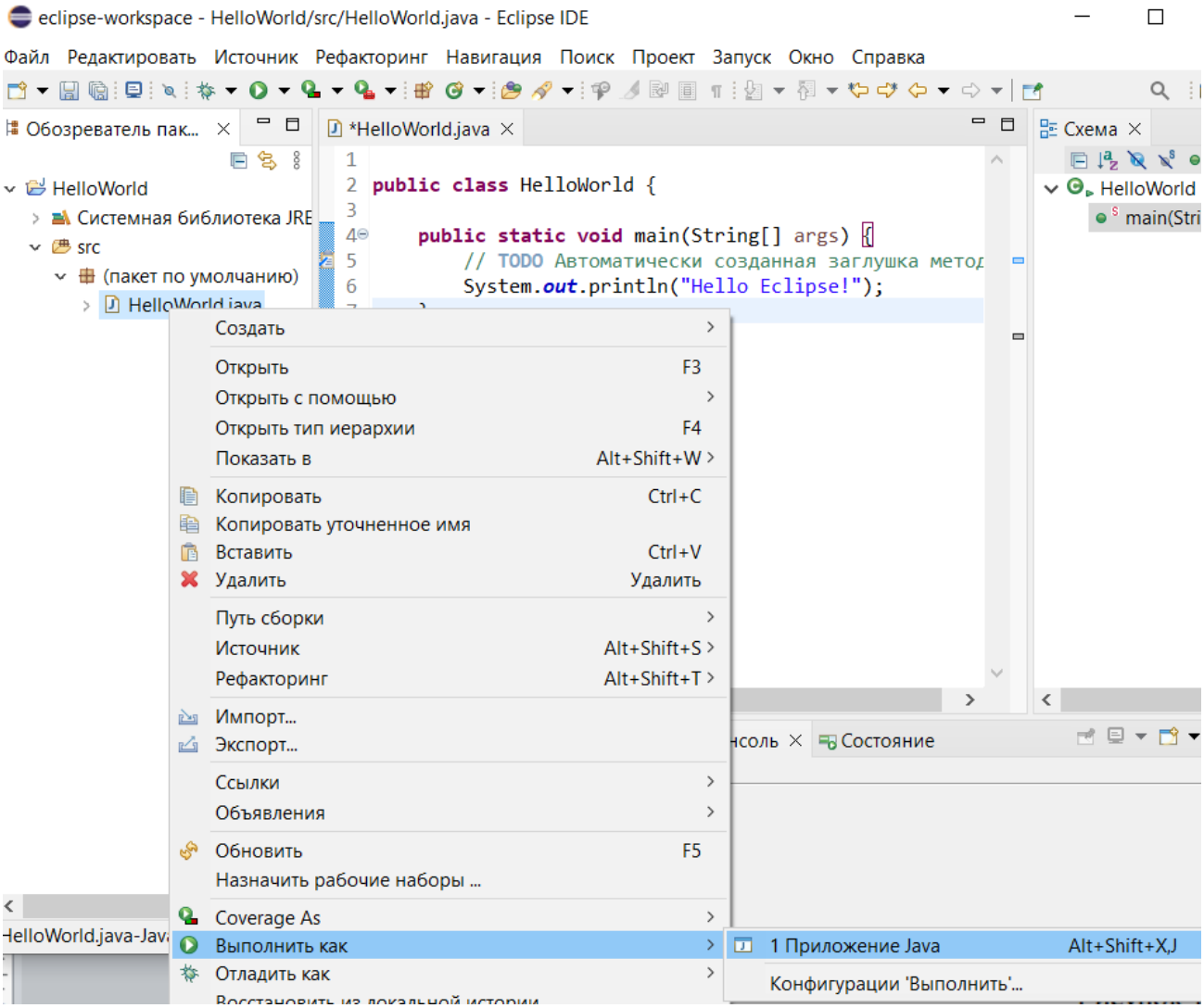

Рисунок 1.9 – Запуск

Результат можете увидеть во вкладке **«Консоль»** (рисунок 1.10).

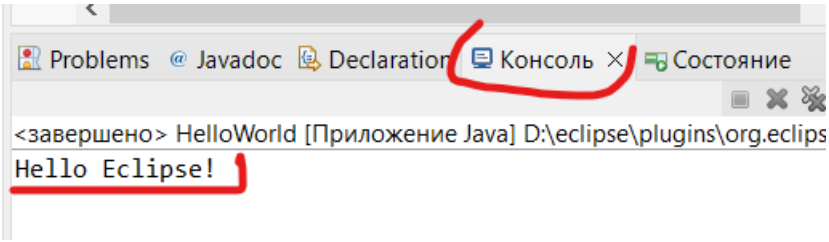

Рисунок 1.10 – Результат выполнения

Любой проект на **Java** состоит из одного или нескольких классов, в этом простейшем проекте только один класс **HelloWorld.** Все, что содержится в классе, записывается в фигурных скобках и составляет тело класса. Класс должен содержать хотя бы один метод main, с него начинается выполнение программы. В описанной выше простейшей программе только один метод и называется он **main**. Метод всегда возвращает (**return**) только одно значение, тип которого обязательно указывается перед именем метода. Метод может и не возвращать никакого значения. Тогда вместо типа возвращаемого значения записывается слово **void.** После имени метода в скобках, через запятую, перечисляются аргументы (**arguments**) – или параметры метода. В примере только один аргумент, его тип – массив, состоящий из строк символов. Имя массива может быть произвольным, но, как правило, используется имя **args**. Перед типом возвращаемого методом значения могут быть записаны его модификаторы (**modifiers**). В примере их два: **public** означает, что этот метод доступен для других классов проекта; **static** обеспечивает возможность использования метода без создания экземпляра 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 аргумента, текстовой константы «`Hello Eclipse`!». Составное имя метода `System.out.println` означает, что в классе System, входящем в Java библиотеку, определена переменная с именем `out` класса `PrintStream`, в котором есть метод `println` ().

Действие метода `println`() заключается в выводе своего аргумента в выходной поток, связанный обычно с выводом на экран текстового терминала. После вывода курсор переходит на начало следующей строки экрана, на что указывает окончание. В составе Java API есть и метод `print`(), оставляющий курсор в конце выведенной строки. Язык Java **различает строчные и прописные буквы**, имена main, Main, MAIN различны с «точки зрения» компилятора Java. В примере важно писать стандартные классы `String, System` с заглавной буквы, а метод `main`() с маленькой.

Соглашения об именах классов.

Свои имена классов можно записывать как угодно, но между Java-программистами заключено соглашение, называемое **«Code Conventions for the Java Programming Language»** (https://habr.com/ru/post/112042/). Вот несколько пунктов этого соглашения:

– имена классов начинаются с прописной буквы; если имя содержит несколько слов, то каждое слово начинается с прописной буквы. Этот стиль называют **CamelCase.** Например, **СтудентыПервогоКурса**;

– имена методов и переменных начинаются со строчной буквы; если имя содержит несколько слов, то каждое следующее слово начинается с прописной буквы. Этот стиль иногда называют **lowerCamelCase**. Например, **weight** (переменная), **вычислениеСуммыМассива** (метод);

– имена констант записываются полностью прописными буквами; если имя состоит из нескольких слов, то между ними ставится знак подчеркивания. Например, **SUMMA**.

1.2.2 Переменные и основные типы данных языка Java

Переменные используются в программе для хранения данных. Любая переменная имеет три базовых характеристики: имя, тип, значение. Имя уникально идентифицирует переменную и позволяет к ней обращаться в программе. Тип описывает, какие величины может хранить переменная. Значение – текущая величина, хранящаяся в переменной на данный момент. Значение может быть указано сразу (инициализация), а в большинстве случаев задание начальной величины можно и отложить.

Объявлять переменную можно в любом месте программы, но до того, как переменная впервые используется. Если объявляется несколько переменных одного типа, то их можно перечислить через запятую после идентификатора типа. Одновременно с объявлением переменной ей можно присвоить значение (инициализировать переменную).

Если переменную объявить и инициализировать с ключевым словом `final`, то такая переменная становится **константой**. Изменить значение константы в процессе выполнения программы нельзя.

```
final double g=9.8;
```

Область доступности переменных определяется блоком, в котором переменная объявлена. Блок, в свою очередь, выделяется парой фигурных скобок { и }.

Базовые типы данных представлены в таблице 1.1.

Таблица 1.1 – Базовые типы данных

Тип данных (название)	Количество байт	Описание	Класс-оболочка
byte	1	Целые числа от –128 до 127	Byte
short	2	Целые числа от –32 768 до 32 767	Short
int	4	Целые числа от –2 147 483 648 до 2147 483 647	Integer
long	8	Целые числа в диапазоне от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807	Long
float	4	Действительные числа $-3{,}4 \times 10^{-38}$ … $3{,}4 \times 10^{38}$	Float
double	8	Действительные числа (двойной точности) $-1{,}7 \times 10^{-308}$ … $1{,}7 \times 10^{308}$	Double
char	2	Символьные значения. Реализуются символы с кодами от 0 до 65 536	Character
boolean	Зависит от виртуальной машины	Логический тип данных. Переменная этого типа может принимать два значения: true и false	Boolean

1.2.3 Операторы Java

Операторы Java можно разделить на четыре группы:

– арифметические,
– логические,
– побитовые,
– операторы сравнения.

Арифметические: +, -, *, /, %, ++, --.
Операторы присваивания: +=, -=, *=, /= и %=.
Операторы инкремента и декремента ++ и --.
Логические операторы: &, &&, |, ||, ^, !.
Операторы сравнения: ==, <, <=, >, >=, !=.
Побитовые операторы: &, |, ^, ~, >>, <<, >>>.
Тернарный оператор, синтаксис вызова этого оператора следующий:

```
условие ? оператор1 : оператор2;
```

Первым операндом указывается некоторое условие (выражение, возвращающее логическое значение). Если условие истинно (значение **true**), тернарный оператор возвращает значение, указанное после вопросительного знака. Если условие ложно (значение **false**), тернарный оператор возвращает значение после двоеточия. Приоритеты операторов указаны в таблице 1.2.

Таблица 1.2 – Приоритеты операторов Java

Приоритет	Операторы
1	*Круглые скобки* ( ), *квадратные скобки* [ ] и оператор *«точка»*
2	*Инкремент* ++, *декремент* --, *побитовая инверсия* ~ и *логическое отрицание* !
3	*Умножение* *, *деление* / и *вычисление остатка* %
4	*Сложение* + и *вычитание* -
5	Побитовые *сдвиги* >>, << и >>>
6	*Больше* >, *больше или равно* >=, *меньше или равно* <= и *меньше* <
7	*Равно* == и *не равно* !=
8	*Побитовое и* &
9	*Побитовое исключающее или* ^
10	*Побитовое или* |
11	*Логическое и* &&
12	*Логическое или* ||
13	*Тернарный оператор* ?:
14	*Присваивание* = и *составные операторы* вида op=

1.2.4 Класс Scanner. Ввод с консоли и вывод данных на консоль

В распоряжении разработчика на языке Java имеется библиотека протестированного кода или готовых программных решений ко многим стандартным задачам, которые многократно решаются программистами. Эта библиотека называется **стандартной библиотекой или библиотекой Java API** (application programming interface – интерфейс прикладного программирования). Классы, входящие в библиотеку, распределены по пакетам («**packages**»). Каждый пакет библиотеки Java API образует отдельный модуль, одно пространство имен (**namespace**), которому соответствует одноименный каталог (папка). Это означает, что все имена классов, представленных в пакете, должны быть уникальны. Количество реализованных классов в пакете не ограниченно. Классы, которые объявляются в пакете, реализуются в файлах с расширением ***.java**.

Кроме классов, пакеты могут включать в себя вложенные подпакеты (**subpackages**) и интерфейсы. Таким образом образуется иерархическая структура библиотеки классов Java. И, соответственно, сохранение пакетов и файлов ***.java** в проекте осуществляется в виде иерархической (древовидной) структуры файловой системы. Компилятор Java в последних версиях имеет более 70 стандартных пакетов. Все пакеты можно найти на сайте Oracle (https://docs.oracle.com/javase/8/docs/api).

Пакет Java – это механизм организации пространства имен (**namespace**): ограниченной области, в которой имена уникальны, но за пределами которой, они не существуют. Формируя независимые комплекты программного обеспечения, пакеты могут распространяться и применяться при разработке приложений в сочетании с другими пакетами. Для определения пакета используется ключевое слово **package**, за которым следует формальное имя пакета, например, **package attr**. Членами пакетов являются взаимосвязанные классы, интерфейсы, вложенные пакеты, а также дополнительные файлы ресурсов (например, графические), используемые в коде классов. Определять пакет не обязательно, но тогда все классы будут находиться в пакете по умолчанию. Классы должны иметь уникальные имена.

Среди всех классов стандартной библиотеки **Java** наиболее широко применяются классы, входящие в пакет `java.lang`. Они составляют основу для всех других классов. Поскольку без классов пакета `java.lang` не может обойтись ни один класс `Java`, модуль компиляции содержит неявное импортирование этого пакета (`import java.lang.*;`). Это означает, что из программы к классам пакета `java.lang` можно всегда обращаться непосредственно по имени, даже если пакет `java.lang` не импортируется явно посредством инструкции `import`.

```
import java.util.Scanner; // импорт класса Scanner
 public class Program { public static void main(String[] args) {
     Scanner in = new Scanner(System.in);
     }
}
```

Директива `import` указывается в самом начале кода, после чего идет имя подключаемого класса (в данном случае класса `Scanner`). В примере подключили только один класс, однако пакет `java.util` содержит еще множество классов. И чтобы не подключать по отдельности каждый класс, мы можем сразу подключить весь пакет:

`import java.util.*;` // импорт всех классов из пакета `java.util`

Теперь мы можем использовать любой класс из пакета `java.util`. В пакете `java.util` есть класс `Scanner`. У него есть методы (действия), которые позволяют работать с вводом и выводом информации в консоль.

Пример:

```
import java.util.Scanner;

public class HelloWorld {
	public static void main(String[] args) {
		// TODO Автоматически созданная заглушка метода
		Scanner in = new Scanner(System.in);
		int a = in.nextInt();
		byte b = in.nextByte();
		String c = in.nextLine();
		String s = in.next();//
		double d = in.nextDouble();
		long e = in.nextLong();
		short f = in.nextShort();
	// вывод в консоль введенных данных
	System.out.println(a + « « + b + « « + c + « « + s + « « + d
				+ « « + e + « « + f);
	}
}
```

Метод `nextInt`() отвечает за ввод целого и присвоения его переменной `a`. Метод `nextByte`() отвечает за ввод байтового числа и присвоения его переменной `b`. Метод `nextLine`() отвечает за ввод одной строки целиком. Метод `next`() считывает строку до первого пробела. Метод `nextLong`() считывает длинное целое число. Метод `nextDouble` () считывает вещественное число. Метод `nextShort`() считывает короткое целое число. Число типа `double` (дробное) вводится через запятую. Иначе выдастся ошибка.

1.2.5 Управляющие инструкции языка Java: if-else, условное выражение, switch

1.2.5.1 Инструкция if-else

Инструкция if-else используется для принятия решения. Ниже представлен ее синтаксис:

```
if (выражение) инструкция_1 else инструкция_2
```

причем **else**-часть может быть опущена. Сначала вычисляется **выражение**, и, если оно истинно, то выполняется **инструкция_1**. Если `выражение` ложно и существует **else**-часть, то выполняется **инструкция_2**. Необходимо отметить, что ***else**-часть всегда относится к ближайшей* `if`*-части*.

1.2.5.2 Условное выражение

Выполнение условного оператора начинается с проверки условия. Если условие истинно (равно **true**), выполняются команды, указанные в фигурных скобках сразу после условия. Если условие ложно (равно **false**), то выполняются команды, размещенные в блоке (выделенном фигурными скобками) после ключевого слова **else**. После выполнения условного оператора управление передается следующей после него команде.

Условное выра*жение*, написанное с помощью *тернарного оператора* «`?:`», представляет собой другой способ записи инструкции `if-else`. В выражении:

```
выражение1 ? выражение2 : выражение3
```

первым вычисляется **выражение1**. Если его значение отлично от нуля, то вычисляется **выражение2** и значение этого выражения становится значением всего условного выражения. В противном случае вычисляется **выражение3**, и его значение становится значением условного выражения.

1.2.5.3 Инструкция switch

Инструкция switch используется для выбора одного из многих путей. Она проверяет, совпадает ли значение выражения с одним из значений, входящих в некоторое множество целых констант, и выполняет соответствующую этому значению ветвь программы:

```
switch(выражение){
case константное_выражение_1: [команда_1]; break;
case константное_выражение_2: [команда_2]; break;
case константное_выражение_n: [команда_n]; break;
[default: команда ];
}
```

1.2.6 Пример программы разветвляющейся структуры

Класс `Math` доступен в программе и без импорта. Статический импорт используется для того, чтобы не указывать название класса при вызове методов. Другими словами, если не использовать статический импорт, то методы из класса пришлось бы вызывать в формате: `Math.sin(), Math.cos()` и `Math.exp()`.

```
import static java.lang.Math.*;
class HelloWorld{
 public static void main(String args[]){
     double a=5;          // Параметры уравнения
     double b=3;
     double c=1;
     double alpha;  // Вспомогательная переменная
     boolean state;       // критерий наличия решений
          alpha=asin(a/sqrt(a*a+b*b)); // Знач. вспомог. перем.
          state=a*a+b*b>=c*c;      // Вычисление критерия:
      // Вывод на экран значений исходных параметров:
     System.out.println(«Уравнение a*cos(x)+b*sin(x)=c»);
     System.out.println(«Параметры:»);
     System.out.println(«a=« + a);
```

```
        System.out.println(«b=« + b);
        System.out.println(«c=« + c);
        System.out.print(«Решение для x: «);
        // Вычисление решения уравнения и вывод на экран:
        System.out.println(state?asin(c/sqrt(a*a+b*b))-alpha:
                                «решений нет!» );
    }
}
```

1.3 Варианты заданий

Составить структурную схему алгоритма и написать программу вычисления функции $z = f(x)$. Варианты функций по указанию преподавателя выбирать из приведенных ниже. Значения параметров a, b и аргумента x вводятся с клавиатуры. Результаты вычислений выводятся на дисплей.

1) $z = \begin{cases} \sin(x) + x^2, если\ x <= a \\ \cos(x) + \sin(x), если\ a < x < b \\ \tan(x) + \cos(x), если\ x >= b \end{cases}$

2) $z = \begin{cases} e^x - \sin(x), если & x <= a \\ \cos(x) + |x|, если & a < x < b \\ x^5, если & x >= b \end{cases}$

3) $z = \begin{cases} e^x, если\ x <= a \\ e^x + \cos(x), если\ a < x < b \\ \tan(x), если\ x >= b \end{cases}$

4) $z = \begin{cases} 1.7 \cdot \sin(x), если & x <= a \\ \cos(x) + x^2, если & a < x < b \\ x^5, если & x >= b \end{cases}$

5) $z = \begin{cases} \ln(x) + \sin(x), если & x <= a \\ \ln(x) + \cos(x), если & a < x < b \\ \tan(x), если & x >= b \end{cases}$

6) $z = \begin{cases} e^x \cdot \sin(x), если & x <= a \\ \tan(x) + x^2, если & a < x < b \\ x^7, если & x >= b \end{cases}$

7) $z = \begin{cases} |x|, если & x <= a \\ |x| + \cos(x), если & a < x < b \\ \tan(x), если & x >= b \end{cases}$

8) $z = \begin{cases} 1.3 + \sin(x), если & x <= a \\ \tan(x) + x^2, если & a < x < b \\ x^7, если & x >= b \end{cases}$

9) $z = \begin{cases} \sin(|x|), если & x <= a \\ \cos(|x|), если & a < x < b \\ \tan(e^x), если & x >= b \end{cases}$

10) $z = \begin{cases} e^x, если & x <= a \\ \cos(x^2), если & a < x < b \\ \tan(x) + 8, если & x >= b \end{cases}$

11) $z = \begin{cases} x^2 + \sin(x), если & x <= a \\ \cos(x^2), если & a < x < b \\ \tan(x), если & x >= b \end{cases}$

12) $z = \begin{cases} \ln(x), если & x <= a \\ 1, если & a < x < b \\ e^x, если & x >= b \end{cases}$

13) $z = \begin{cases} |x| + \sin(x), если & x <= a \\ \cos(|x|), если & a < x < b \\ \tan(x), если & x >= b \end{cases}$

14) $z = \begin{cases} \ln(x) \cdot \sin(x), если & x <= a \\ x^2 \cdot \cos(x), если & a < x < b \\ x^5, если & x >= b \end{cases}$

15) $z=\begin{cases}\ln(x)+|x|, если & x<=a\\ x^3\cdot\cos(x), если & a<x<b\\ x^4, если & x>=b\end{cases}$

16) $z=\begin{cases}e^x+|x|, если & x<=a\\ \tan(x)+x^2+|x|, если & a<x<b\\ x^7, если & x>=b\end{cases}$

17) $z=\begin{cases}e^x-\sin(x), если & x<=a\\ \cos(x-2.3), если & a<x<b\\ x^5, если & x>=b\end{cases}$

18) $z=\begin{cases}e^x\cdot\sin(x)-|x|, если & x<=a\\ \tan(x)+x^2+|x|, если & a<x<b\\ x^7, если & x>=b\end{cases}$

19) $z=\begin{cases}e^x\cdot\sin(x), если & x<=a\\ \cos(x)+x^2, если & a<x<b\\ x^5, если & x>=b\end{cases}$

20) $z=\begin{cases}e^x-\sin(x), если & x<=a\\ \tan(x)\cdot x^2, если & a<x<b\\ x^7+|x|, если & x>=b\end{cases}$

21) $z=\begin{cases}\log_{10}x, если & x<=a\\ e^x, если & a<x<b\\ e^x/(3+\sin(x)), если & x>=b\end{cases}$

22) $z=\begin{cases}1.3+\sin(x), если & x<=a\\ e^x/(3+\sin(x)), если & a<x<b\\ \cos(x)+x^2, если & x>=b\end{cases}$

23) $z=\begin{cases}e^x/(3+\sin(x)), если & x<=a\\ \cos(x)+x^2, если & a<x<b\\ 1.3+\sin(x), если & x>=b\end{cases}$

24) $z=\begin{cases}4, если: x<=a\\ \cos(x)+x^2, если: a<x<b\\ \cos(x-2.3), если: x>=b\end{cases}$

25) $z=\begin{cases}e^x/(3+\sin(x)), если & x<=a\\ \tan(x)+x^2, если & a<x<b\\ x^7, если & x>=b\end{cases}$

26) $z=\begin{cases}\cos(x), если & x<=a\\ \ln(x)\cdot\sin(x), если & a<x<b\\ 0, если & x>=b\end{cases}$

27) $z=\begin{cases}e^x\cdot\sin(x), если & x<=a\\ \tan(|x|)+x^2, если & a<x<b\\ x^7, если & x>=b\end{cases}$

28) $z=\begin{cases}e^x/(3+\sin(x)), если & x<=a\\ \ln(x)+x^2, если & a<x<b\\ 1+\sin(-x), если & x>=b\end{cases}$

29) $z=\begin{cases}e^x\cdot\sin(x), если & x<=a\\ \tan(x)+\ln(x), если & a<x<b\\ x^7, если & x>=b\end{cases}$

30) $z=\begin{cases}x-2\cos^2(x), если & x<=a\\ \ln(x)\cdot\sin(x), если & a<x<b\\ 1.3+\sin(x), если & x>=b\end{cases}$

1.4 Порядок выполнения работы

1.4.1 Проработать необходимый теоретический материал, опираясь на сведения и указания, представленные в предыдущем пункте.

1.4.2 Ответить на контрольные вопросы.

1.4.3 Внимательно изучить постановку задачи.

1.4.4 Выполнить анализ *области определения* и *области значений* вычисляемой функции z.

1.4.5 Разработать структурную схему алгоритма решения задачи.

1.4.6 Разработать *тестовые примеры*, которые должны включать, по крайней мере, по два значения аргумента из каждого интервала кусочно-заданной функции z.

Тестовые примеры должны также отражать поведение функции z в граничных точках.

1.4.7 Написать программу вычисления функции z. Программа *обязательно* должны содержать комментарии.

1.4.8 Выполнить *тестирование и отладку* написанных программ, используя разработанные тестовые примеры. Особое внимание следует обратить на значения функции z в граничных точках.

1.4.9 Подготовить отчет по работе.

1.5 Содержание отчета

Цель работы; постановка задачи и вариант задания; математическое обоснование задачи (включает анализ области определения и области значений функции, подлежащей вычислению); структурная схема алгоритма решения задачи; тестовые примеры; текст программы; результаты тестирования и отладки программ; выводы.

1.6 Контрольные вопросы

1.6.1 Перечислите и охарактеризуйте основные концепции объектно-ориентированного программирования.

1.6.2 Перечислите и охарактеризуйте спецификаторы доступа.

1.6.3 Перечислите и охарактеризуйте базовые типы языка `Java`.

1.6.4 Что такое главный метод?

1.6.5 Что такое приведение типа? Приведите пример.

1.6.6 Приведите пример объявления и инициализации переменной и константы на Java.

1.6.7 Перечислите и охарактеризуйте арифметические операторы.

1.6.8 Перечислите и охарактеризуйте операторы отношения.

1.6.9 Перечислите и охарактеризуйте логические операторы.

1.6.10 Перечислите и охарактеризуйте операторы присваивания.

1.6.11 Опишите средства текстового ввода-вывода языка `Java`.

1.6.12 Как задаются комментарии на `Java`?

1.6.13 Опишите инструкцию `if-else`.

1.6.14 Что такое условное выражение (тернарный оператор)? Приведите пример.

1.6.15 Опишите инструкцию `switch`. Приведите пример.

2. ЛАБОРАТОРНАЯ РАБОТА №2 «ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ НА JAVA»

2.1 Цель работы

Получение навыков программирования алгоритмов циклической структуры на языке Java. Исследование эффективности применения различных видов циклов в задаче табулирования функции.

2.2 Краткие теоретические сведения

2.2.1 Циклы на языке Java

Цикл представляет собой участок программы, повторяемый многократно. На языке Java имеется три разновидности циклов: **while**, **do-while** и **for**.

2.2.1.1 Цикл while

Формат цикла с предусловием

```
while (выражение) команда;
```

После ключевого слова **while** в круглых скобках указывается условие. Оно проверяется в начале выполнения оператора цикла **while**. Если условие истинно, то выполняются команды в теле оператора цикла (если их несколько, то они заключаются в фигурные скобки). После этого снова проверяется условие. Если оно истинно, то вновь выполняются команды и проверяется условие, и так далее. Работа оператора цикла завершается, если при очередной проверке условия оно окажется ложным.

2.2.1.2 Цикл do-while

Цикл с постусловием и имеет вид:

```
do {команда;
} while (выражение);
```

Выполнение оператора начинается с блока команд, размещенных в фигурных скобках после ключевого слова **do**. Затем проверяется условие, указанное в круглых скобках после ключевого слова **while**. Если условие истинно, то снова выполняются команды и затем проверяется условие, и так далее. Тело цикла хотя бы один раз будет выполнено!

2.2.1.3 Цикл for

Цикл с параметром имеет следующий формат:

```
for (выражение1; выражение2; выражение3) команда;
```

Выражения разделяются точкой с запятой.

Выражение 1 – инициализация;

выражение 2 – условие окончания цикла;

выражение 3 – обновление.

Сначала выполняется инициализация первого выражения (только один раз в самом начале работы оператора цикла). Затем проверяется условие во втором выражении. Если оно истинно (значение true), то выполняются команды в теле оператора

цикла (команды в фигурных скобках). Далее выполняется обновление – третье выражение в круглых скобках, и снова проверяется условие из второго выражения. Если условие истинно, то выполняются команды в теле оператора цикла и третье выражение, и затем снова проверяется условие, и так далее. Если при проверке условия окажется, что оно ложно (значение false), то выполнение оператора цикла на этом завершается.

2.2.1.4 Инструкции break и continue

Инструкции `break` и `continue` изменяют поток управления. Когда инструкция `break` выполняется в инструкциях `switch`, `while`, `do-while` или `for`, происходит немедленный выход из соответствующей инструкции, и управление передается в точку, расположенную сразу за инструкцией. Обычное назначение инструкции `break` – досрочно прервать цикл или пропустить оставшуюся часть инструкции `switch`. Необходимо заметить, что инструкция `break` вызывает немедленный выход из *самого внутреннего* из объемлющих ее циклов.

Инструкция `continue` в циклах `while`, `do-while` или `for` вызывает пропуск оставшейся части тела цикла, после чего начинается выполнение следующей итерации цикла. В циклах `while` и `do-while` после выполнения инструкции `continue` осуществляется проверка условия продолжения цикла. В цикле `for` после выполнения инструкции `continue` выполняется *выражение приращения* (`выражение3`), а затем осуществляется проверка условия продолжения цикла (`выражение2`). Таким образом, инструкция `continue` вынуждает ближайший объемлющий ее цикл начать следующий шаг итерации.

2.3 Варианты заданий

Вычислить и вывести на экран в виде таблицы значения функции $z = f(x)$ на интервале от $x_{нач}$ до $x_{кон}$ с шагом Δx. Таблицу снабдить заголовком и шапкой. Вид функции z выбирать в соответствии с вариантами задания к лабораторной работе №1 настоящих методических указаний. Значения параметров a, b, а также $x_{нач}$, $x_{кон}$ и Δx вводятся с клавиатуры.

2.4 Порядок выполнения работы

2.4.1 Проработать необходимый теоретический материал, опираясь на сведения и указания, представленные в предыдущем пункте, а также на лекционный материал.

2.4.2 Ответить на контрольные вопросы.

2.4.3 Внимательно изучить постановку задачи.

2.4.4 Выполнить анализ *области определения* и *области значений* вычисляемой функции z.

2.4.5 Разработать структурную схему алгоритма решения задачи.

2.4.6 Разработать *тестовые примеры* (таблицы значений функции). При разработке тестовых примеров целесообразно использовать *отлаженную* программу из предыдущей лабораторной работы для вычисления значений функции z.

2.4.7 Написать *два* варианта программы табулирования функции z, используя циклы **while** и **for**. Программы *обязательно* должны содержать комментарии.

2.4.8 Выполнить *тестирование и отладку* написанных программ, используя разработанные тестовые примеры.

2.4.9 Подготовить отчет по работе.

2.5 Содержание отчета

Цель работы; постановка задачи и вариант задания; математическое обоснование задачи (включает анализ области определения и области значений функции, подлежащей вычислению); структурная схема алгоритма решения задачи; тестовые примеры; тексты программ; результаты тестирования и отладки программ; выводы.

2.6 Контрольные вопросы

2.6.1 Что называют циклом?
2.6.2 Перечислите типы циклов в языке Java.
2.6.3 Охарактеризуйте цикл с предусловием.
2.6.4 Охарактеризуйте цикл с постусловием.
2.6.5 Охарактеризуйте цикл **while**.
2.6.6 Охарактеризуйте цикл **do-while**.
2.6.7 Охарактеризуйте цикл **for**.
2.6.8 Опишите инструкции **break** и **continue**.
2.6.9 В каких случаях целесообразно применять оператор запятая?
2.6.10 Какой тип цикла лучше использовать в задаче табулирования функции? Почему?

3 ЛАБОРАТОРНАЯ РАБОТА №3 «ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ ОБРАБОТКИ ОДНОМЕРНЫХ МАССИВОВ НА JAVA»

3.1 Цель работы

Изучить способы представления массивов в памяти ЭВМ, получить практические навыки реализации алгоритмов обработки одномерных массивов на языке Java.

3.2 Краткие теоретические сведения

Для выполнения лабораторной работы необходимо изучить особенности представления и обработки одномерных массивов на языке Java, а также алгоритмы сортировки массивов методами прямого обмена, вставки и прямого выбора.

3.2.1 Одномерные массивы

Одномерный массив – это набор объектов одинакового типа, расположенных в **последовательной** группе ячеек памяти, доступ к которым осуществляется **по индексу**. В языке Java одномерные массивы описываются следующим образом:

```
тип[] имя=new тип[размер];
```

Например, командой

```
int[] nums=new int[20];
```

объявляется целочисленный массив **nums** из 20 элементов, или можно записать так:

```
int[] nums;
nums=new int[20];
double[] x1={1.1, 2.2, 3.3};
double x2[];
     x2=new double[]{-1.1, -2.2, -3.3};
float[] x3={1.1f, 2.2f, 3.3f};
char[] x4={'1','2','3', '4'};
boolean[] x5={true, false, false, true};
```

Индексация элементов массива начинается ***с нуля***. Таким образом, обращение к первому элементу массива **nums** будет иметь вид **nums[0]**. Если в массиве 20 элементов, то последний элемент массива имеет индекс 19, то есть инструкция обращения к элементу выглядит как **nums[19].**

Длину массива можно узнать с помощью свойства **length**:

```
int n=nums.length;
```

Тогда ссылка на последний элемент массива может быть записана как **`nums[nums.length-1]`.**

3.2.2 Поиск в массивах

Напишем программу, которая для целочисленного массива из n элементов определяет индекс минимального элемента.

```
import java.util.Scanner;
public class Demo1 {
     public static void main(String[] args) {
               int n, i;
               Scanner input = new Scanner(System.in);
               System.out.print(«Введите количество элементов: «);
               n = input.nextInt();
               int x[];
               x = new int[n];

               System.out.println(«Введите элементы массива:»);
               for (i = 0; i < x.length; i++)
                    x[i]=input.nextInt();

               System.out.println(«Массив:»);
               for (i = 0; i < x.length; i++)
                    System.out.print(« « + x[i]);

               int imin=0;
               for (i = 1; i < x.length; i++)
                    if (x[i]<x[imin]) imin=i;
               System.out.println(«\nimin=«+imin+»  min=«+x[imin]);
          }
}
```

3.3 Варианты заданий

Значение константы N (размера массива) студент выбирает самостоятельно.

Вариант 1

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) сумму положительных элементов массива;

2) произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.

Упорядочить элементы массива по убыванию, используя алгоритм сортировки методом прямого обмена.

Вариант 2

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) сумму отрицательных элементов массива;

2) произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.

Упорядочить элементы массива по убыванию, используя алгоритм сортировки методом вставки.

Вариант 3

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) произведение элементов массива с четными номерами;

2) сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Преобразовать массив таким образом, чтобы сначала располагались все неотрицательные элементы, а потом все отрицательные. Выполнить сортировку каждой части массива по возрастанию методом прямого выбора.

Вариант 4

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) сумму элементов массива с нечетными номерами;

2) сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Заменить все элементы, модуль которых не превышает 1, на 1 и упорядочить элементы массива по возрастанию, используя алгоритм сортировки методом прямого обмена.

Вариант 5

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) максимальный элемент массива;

2) сумму элементов массива, расположенных до последнего положительного элемента.

Заменить все элементы, модуль которых находится внутри отрезка `[a, b]` на число `a` и упорядочить элементы массива по возрастанию, используя алгоритм сортировки методом вставки. Значение `a` и `b` вводит пользователь.

Вариант 6

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) минимальный элемент массива;

2) сумму элементов массива, расположенных между первым и последним положительными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, большие 1, а потом – все остальные. Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом прямого выбора.

Вариант 7

В одномерном массиве, состоящем из *N* целых элементов, вычислить:

1) номер максимального элемента массива;

2) произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие на нечетных позициях, а во второй половине – элементы, стоявшие на четных позициях. Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом прямого обмена.

Вариант 8

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) номер минимального элемента массива;

2) сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом – все остальные. Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом вставки.

Вариант 9

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) максимальный по модулю элемент массива;

2) сумму элементов массива, расположенных между первым и вторым положительными элементами.

Преобразовать массив таким образом, чтобы элементы, большие 1, располагались после всех остальных. Упорядочить каждую часть массива по убыванию, используя алгоритм сортировки методом прямого выбора.

Вариант 10

В одномерном массиве, состоящем из *N* целых элементов, вычислить:

1) минимальный по модулю элемент массива;

2) сумму модулей элементов массива, расположенных после первого элемента, равного нулю.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине – элементы, стоявшие в нечетных позициях. Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом прямого обмена.

Вариант 11

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) номер минимального по модулю элемента массива;

2) сумму модулей элементов массива, расположенных после первого отрицательного элемента.

Сжать массив, удалив из него все элементы, величина которых находится внутри отрезка `[a, b]`. Упорядочить элементы массива по возрастанию, используя алгоритм сортировки методом вставки. Значение `a` и `b` вводит пользователь.

Вариант 12

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) номер максимального по модулю элемента массива;

2) сумму элементов массива, расположенных после первого положительного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит внутри отрезка `[a, b]`, а потом – все остальные. Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом прямого обмена. Значение `a` и `b` вводит пользователь.

Вариант 13

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) количество элементов массива, лежащих в диапазоне от `A` до `B`;

2) сумму элементов массива, расположенных после максимального элемента.

Упорядочить элементы массива по убыванию модулей элементов методом вставки. Значение A и B вводит пользователь.

Вариант 14

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) количество нулевых элементов массива;

2) сумму элементов массива, расположенных после минимального элемента.

Упорядочить элементы массива по возрастанию модулей элементов методом прямого обмена.

Вариант 15

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) количество элементов массива, больших C (значение C вводит пользователь);

2) произведение элементов массива, расположенных после максимального по модулю элемента.

Преобразовать массив таким образом, чтобы сначала располагались все неотрицательные элементы, а потом – все отрицательные. Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом вставки.

Вариант 16

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) количество отрицательных элементов массива;

2) сумму модулей элементов массива, расположенных после минимального по модулю элемента.

Заменить все отрицательные элементы массива их квадратами и упорядочить элементы массива по возрастанию, используя алгоритм сортировки методом прямого обмена.

Вариант 17

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) количество положительных элементов массива;

2) сумму элементов массива, расположенных после последнего элемента, равного нулю.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых не превышает 1, а потом – все остальные. Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом прямого выбора.

Вариант 18

В одномерном массиве, состоящем из *N* целых элементов, вычислить:

1) количество элементов массива, меньших C (значение C вводит пользователь);

2) сумму положительных элементов массива, расположенных после последнего отрицательного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все четные элементы, а потом – все нечетные. Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом вставки.

Вариант 19

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) произведение отрицательных элементов массива;

2) сумму положительных элементов массива, расположенных до максимального элемента. Изменить порядок следования элементов в массиве на обратный.

Вариант 20

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) произведение положительных элементов массива;

2) сумму элементов массива, расположенных до минимального элемента.

Упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах, методом вставки.

Вариант 21

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) сумму элементов массива, значения которых превышают 3;

2) произведение элементов массива, расположенных до максимального и после минимального элементов.

Упорядочить элементы массива по возрастанию, используя алгоритм сортировки методом прямого обмена.

Вариант 22

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) сумму элементов массива, значения которых превышают 1;

2) произведение элементов массива, расположенных после максимального по модулю и до минимального по модулю элемента.

Упорядочить элементы массива по убыванию, используя алгоритм сортировки методом вставки.

Вариант 23

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) произведение элементов массива с четными номерами, значения которых превышают 1;

2) сумму элементов массива, расположенных до первого и после последнего нулевого элементов.

Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом все отрицательные. Выполнить сортировку каждой части массива методом прямого выбора по возрастанию.

Вариант 24

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) сумму элементов массива с нечетными номерами, значения которых превышают 0;

2) сумму элементов массива, расположенных до первого и после последнего отрицательного элемента.

Удалить все элементы, модуль которых не превышает 1, и упорядочить элементы массива по возрастанию, используя алгоритм сортировки методом прямого обмена.

Вариант 25

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) максимальный элемент массива;

2) сумму элементов массива, расположенных после последнего положи-

тельного элемента.

Заменить все элементы, модуль которых находится внутри отрезка [a, b] на число b и упорядочить элементы массива по возрастанию, используя алгоритм сортировки методом прямого обмена. Значение a и b вводит пользователь.

Вариант 26

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) минимальный элемент массива;

2) сумму элементов массива, расположенных до первого и после последнего положительного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, больше 1, а потом – все остальные. Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом прямого обмена.

Вариант 27

В одномерном массиве, состоящем из *N* целых элементов, вычислить:

1) номер максимального элемента массива;

2) произведение элементов массива, расположенных после первого нулевого элемента.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие на нечетных позициях, а во второй половине – элементы, стоявшие на четных позициях. Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом прямого обмена.

Вариант 28

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) номер минимального элемента массива;

2) сумму элементов массива, расположенных после первого отрицательного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом – все остальные. Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом вставки.

Вариант 29

В одномерном массиве, состоящем из *N* вещественных элементов, вычислить:

1) максимальный по модулю элемент массива;

2) сумму элементов массива, расположенных после второго положительного элемента.

Преобразовать массив таким образом, чтобы элементы, больше 1, располагались после всех остальных. Упорядочить каждую часть массива по убыванию, используя алгоритм сортировки методом прямого выбора.

Вариант 30

В одномерном массиве, состоящем из *N* целых элементов, вычислить:

1) минимальный по модулю элемент массива;

2) сумму модулей элементов массива, расположенных после второго элемента, равного нулю.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие на нечетных позициях, а во второй половине – элемен-

ты, стоявшие на четных позициях. Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом прямого обмена.

3.4 Порядок выполнения работы

3.4.1 Разработать структурную схему алгоритма решения задачи.

3.4.2 Написать соответствующую программу.

3.4.3 Составить тестовые примеры, следуя указаниям, изложенным в разделе 3.2.

3.4.4 Выполнить отладку программы для всех тестовых примеров.

3.5 Содержание отчета

Цель работы, вариант задания, структурная схема алгоритма, текст программы, тестовые примеры, выводы.

3.6 Контрольные вопросы

3.6.1 Как на языке Java описываются одномерные массивы?

3.6.2 Каковы особенности инициализации массива при его объявлении?

3.6.3 Как осуществляется обращение к отдельным элементам массива?

3.6.4 Напишите фрагмент программы, выполняющий поиск заданного элемента в массиве.

3.6.5 Напишите фрагмент программы, выполняющий поиск минимального элемента в массиве.

3.6.6 Объясните алгоритм сортировки методом пузырька.

3.6.7 Объясните алгоритм сортировки методом вставки.

3.6.8 Объясните алгоритм сортировки методом прямого выбора.

3.6.9 Напишите программу сортировки методом пузырька.

3.6.10 Напишите программу сортировки методом вставки.

3.6.11 Напишите программу сортировки методом прямого выбора.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гуськова, О. И. Объектно ориентированное программирование в Java : учебное пособие / О. И. Гуськова. — Москва : МПГУ, 2018. — 240 с. — ISBN 978-5-4263-0648-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/122311.

2. Березовская, Ю. В. Основы программирования на JAVA: лабораторный практикум : учебно-методическое пособие / Ю. В. Березовская. — Архангельск : САФУ, 2016. — 113 с. — ISBN 978-5-98450-442-3.

3. Макаров, Е. М. Элементы двумерной графики в Java : учебно-методическое пособие / Е. М. Макаров. — Нижний Новгород : ННГУ им. Н. И. Лобачевского, 2017. — 56 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/152985.

4. Хабитуев, Б. В. Программирование на языке Java: практикум : учебное пособие / Б. В. Хабитуев. — Улан-Удэ : БГУ, 2020. — 94 с. — ISBN 978-5-9793-1548-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/171791.

5. Пруцков, А. В. Язык программирования Java. Введение в курс: объектно-ориентированное программирование : учебное пособие / А. В. Пруцков. — Рязань : РГРТУ, 2016. — 56 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/168308.

6. Пруцков, А. В. Язык программирования Java. Введение в курс: операторы и типы данных : учебное пособие / А. В. Пруцков. — Рязань : РГРТУ, 2016. — 72 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/168307.

Заказ № _____ от «_____» _____2021г. Тираж _____экз.
Изд-во СевГУ