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 1
Основы работы с инструментальной системой BlueJ. Создание простых программ с линейной структурой для обработки данных простых типов

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 32.973-018.1я7
П 784

П 783 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 1. Основы работы с инструментальной системой BlueJ. Создание простых программ с линейной структурой для обработки данных простых типов / 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, О. В. Ченгарь. – Севастополь : [Изд-во СевГУ], 2020. – 58 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 простых программ с линейной структурой на языке Java в системе программирования BlueJ. 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 32.973-018.1я7

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

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 №1. ВЫПОЛНЕНИЕ ПРОСТЫХ ПРОГРАММ В BLUEJ

2.1. Цель работы

Освоить основы применения BlueJ – подготовку текста программы, компиляцию программы, исправление ошибок и просмотр результатов.

2.2. Постановка задачи

Разработать простейшую линейную программу, согласно варианту задания, научиться запускать программу и контролировать выводимый текст (результат работы программы).

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.5-24).

2.4. Краткие теоретические сведения

Составление программы – процесс, направленный на то, чтобы заставить компьютер выполнить определенную работу. Указание компьютеру должно быть представлено как последовательность инструкций (план, программа). Проверка того, что данный набор инструкций выполняется правильно, состоит в сравнении желаемого результата с тем, который дает компьютер.

BlueJ – система для разработки программ на языке Java, созданная специально для обучения (http://www.bluej.org). Для решения отдельной задачи BlueJ создает **проект**. Проект – это каталог, в котором находится файл с текстом программы и ряд других файлов. Имя проекта задает программист.

Набор и коррекция программы производится при помощи встроенного текстового редактора. Файл с программой имеет расширение java. Компилятор (javac) переводит программу в команды виртуальной Java-машины (JVM) и сохраняет в файле с расширением class. Такой файл будет выполнять Java-машина, и программа сможет проделать работу, ради которой она создана (рисунок 2.1).

Для составления программ студент должен владеть базовыми понятиями программирования, перечисленными ниже.

Идентификаторы используются в качестве имен классов, методов и переменных. Идентификатор может быть любой последовательностью букв верхнего и нижнего регистра (в том числе, кириллических), чисел или символов подчеркивания и знака коммерческого S ($). Он не должен начинаться с цифры, чтобы не вступать в конфликт с числовой константой. Язык Java

чувствителен к регистру, поэтому идентификатор VALUE, например, отличается от идентификатора Value.

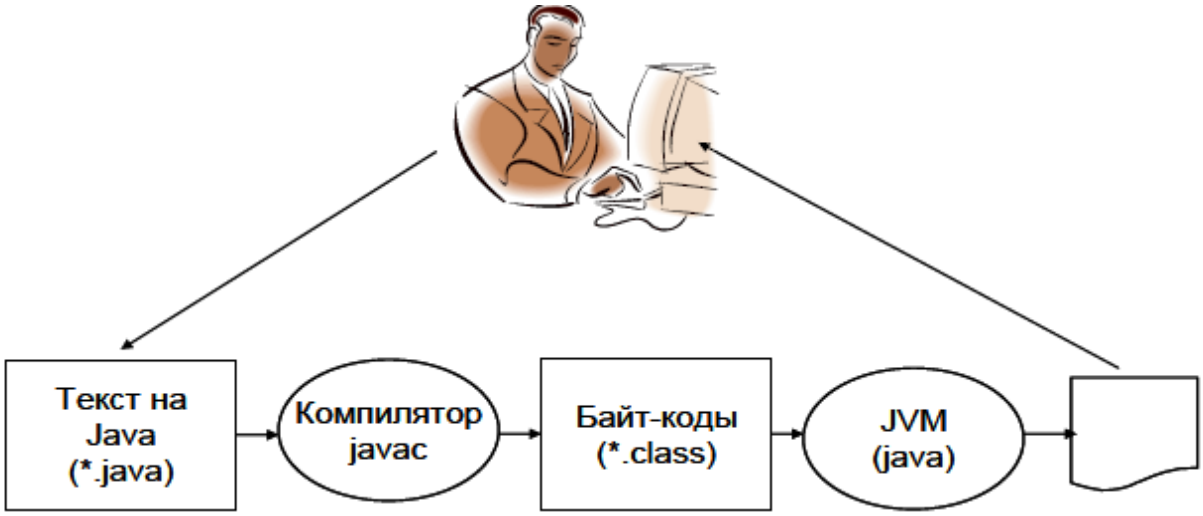

Рисунок 2.1. Подготовка и выполнение java-программы.

Переменная – это именованная область памяти, в которой программа может установить некоторое значение. Значение переменной может изменяться во время выполнения программы.

Переменная определяется комбинацией идентификатора (имени), типа и необязательного инициализатора. Переменная должна быть объявлена перед ее использованием.

Синтаксис объявления переменной:

type identifier [=value][, identifier [=value]…];
type – один из типов Java, имя класса или интерфейса,
identifier – имя переменной,
value – литерал (значение подходящего типа).
Примеры:
int a, b, c;
int d = 3, e, f = 5;

Выражение – комбинация операндов и операций, задающая порядок вычисления некоторого значения (на основе приоритетов операций).

Операнд в простейшем случае является константой (литералом) или идентификатором (переменной). В общем случае каждый операнд выражения также представляет собой выражение, имеющее некоторое значение (в выражениях можно использовать скобки для изменения порядка действий).

Операция определяет действие, выполняемое над операндами. Возвращает некоторое значение.

Оператор − это некоторая конструкция, присущая данному конкретному языку, изменяющая состояние памяти компьютера, но ничего не возвращающая.

Замечание: *Не стоит путать два таких понятия как оператор и операция. Главное их отличие состоит в том, что операция возвращает значение, а оператор нет.*

Оператор присваивания предписывает вычисление выражения, находящегося правее знака (=) и присвоение полученного значения переменной, находящейся левее знака (=).

2.5. Порядок выполнения работы

Часть 1. Создание и выполнение Java –программы.

Основное окно системы BlueJ показано на рисунке 2.2. Окно содержит главное меню системы, управляющие кнопки, окно проекта, стенд объектов, окно команд (окно кода) и индикатор работы виртуальной машины Java.

Запуск BlueJ происходит после щелчка по иконке BlueJ. Дождитесь сообщения «Создается виртуальная машина…Готово». Выберите в меню BlueJ **Проект--Новый**. Введите имя проекта: «Фамилия_Группа_Lab1a».

Щелкните по кнопке **Новый класс**. Введите имя класса «Test1a». Убедитесь, что тип класса – Класс.

Двойным щелчком по иконке класса Test1a в окне проекта откройте текстовый редактор. При создании класса система BlueJ использует заготовку. Удалите из созданного класса весь код, нажав клавиши Ctrl-A (выделить все) и Delete.

Окно редактора с нужным текстом показано на рисунке 2.3.

Введите текст, приведенный на рисунке 2.3. Проверьте правильность набора (оператор import (импортирование из библиотеки Java), точки с запятой после каждого оператора). Проверьте парность скобок { и }. Помните, что в Java большие и маленькие буквы различаются.

Правильность набора текста программы проверяется визуально и при помощи компилятора. Компилятор, анализируя программу по строкам сверху вниз, проверяет соответствие языку Java. При несоответствии выдается сообщение об обнаруженной ошибке. Подсвечивается строка программы, на которой компилятор прекратил работу. Обычно ошибка находится в этой строке или выше. Запомните: компилятор не проверяет правильность алгоритма, он проверяет только соответствие текста правилам языка Java! Поэтому возможно, что программа, которая компилируется без ошибок, будет работать неверно.

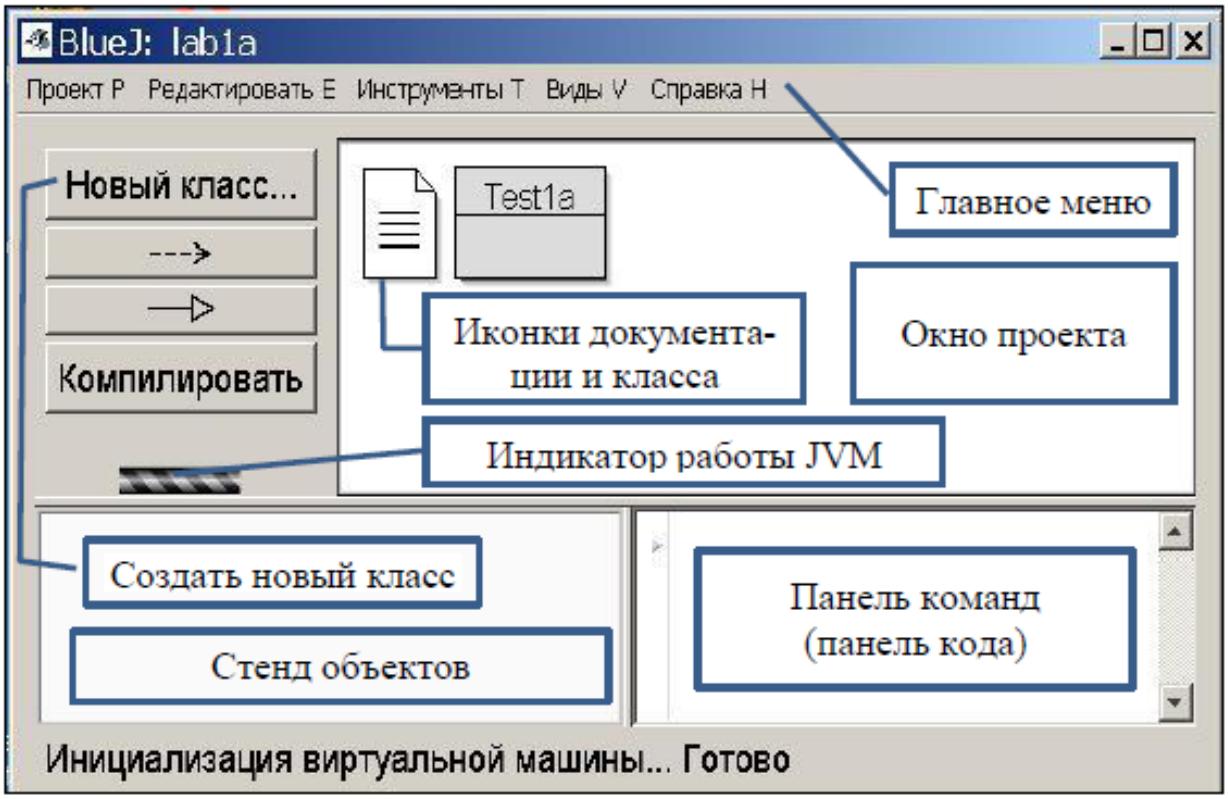

Рисунок 2.2 – Основное окно системы

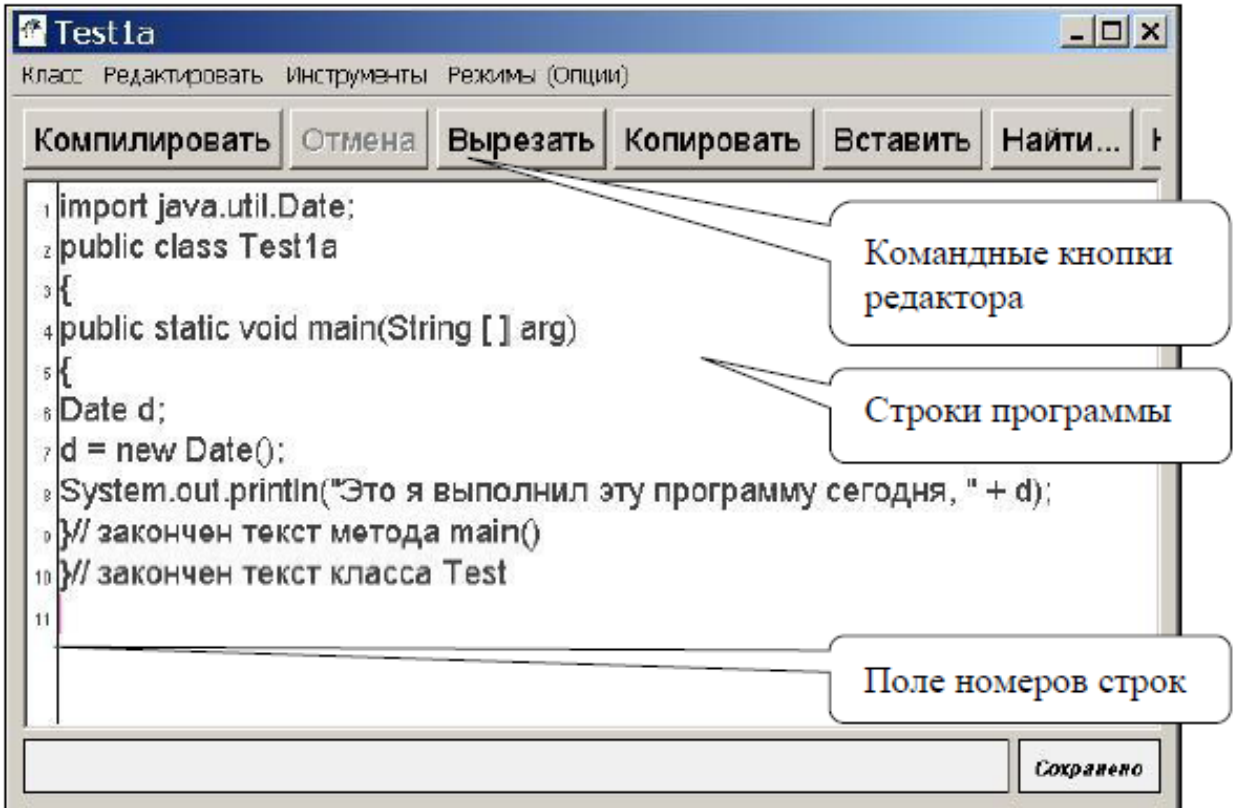

Рисунок 2.3 – Окно текстового редактора

Щелкните по кнопке **Компилировать**. Исправьте синтаксические ошибки, если они есть (ориентируйтесь на сообщения в нижней части окна). Если ошибок нет, можно выйти из редактора (кнопка Закрыть).

В окне проекта штриховка иконки класса показывает, что он не скомпилирован. Если какие-то классы проекта не скомпилированы, кнопкой Компилировать можно вызвать их компиляцию.

Щелкните правой кнопкой мышки по иконке класса, чтобы вызвать контекстное меню (рисунок 2.4). Выберите в меню метод **void main(String[ ] arg)**.

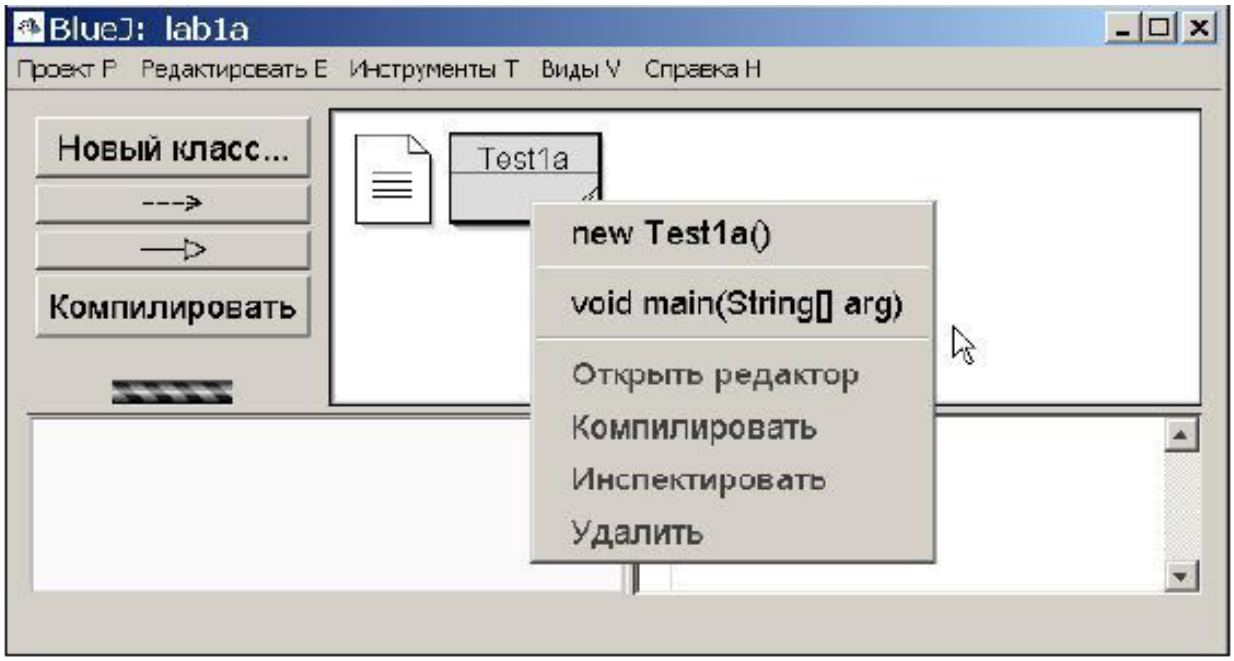

Рисунок 2.4. Окно проекта. Открыто контекстное меню класса

Появится окно диалога, в котором можно задать аргумент arg метода main. Но сейчас аргумент не нужен, просто щелкните по OK.

Запустится метод main, при этом индикатор работы VM начнет вращаться (вращение можно заметить только на медленных машинах).

Результат работы программы – текст, указанный в методе println и дата – будут выведены в окно терминала (рисунок 2.5). (Если окна терминала нет, в главном окне выберите Виды – Показать терминал).

Опции окна терминала позволяют выполнить ряд действий с содержимым терминала: очистить окно терминала, скопировать текст в буфер обмена или в файл, зафиксировать порядок вызова методов и т.д.

Часть 2. Внесение изменений в программу.

Измените строковый литерал (текст в двойных кавычках после println) так, чтобы выводились Ваши имя и фамилия. Обратите внимание на возможности, кото-рые предоставляет текстовый редактор. Кнопки редактора (**Отмена**, **Вырезать**, **Копировать**, **Вставить**, **Найти**) соответствуют типовым действиям

по коррекции текстов. Эти же (и ряд других) действия скрыты под пунктами главного меню в верхней части окна. Многим командам редактора соответствуют "быстрые" комбинации клавиш. Например, Вырезать, Копировать и Вставить – это широко известные Ctrl+X, Ctrl+C, Ctrl+V, соответствующие одновременному нажатию клавиши Ctrl и буквенных клавиш X, C или V.

Откомпилируйте программу и проанализируйте результаты ее выполнения.

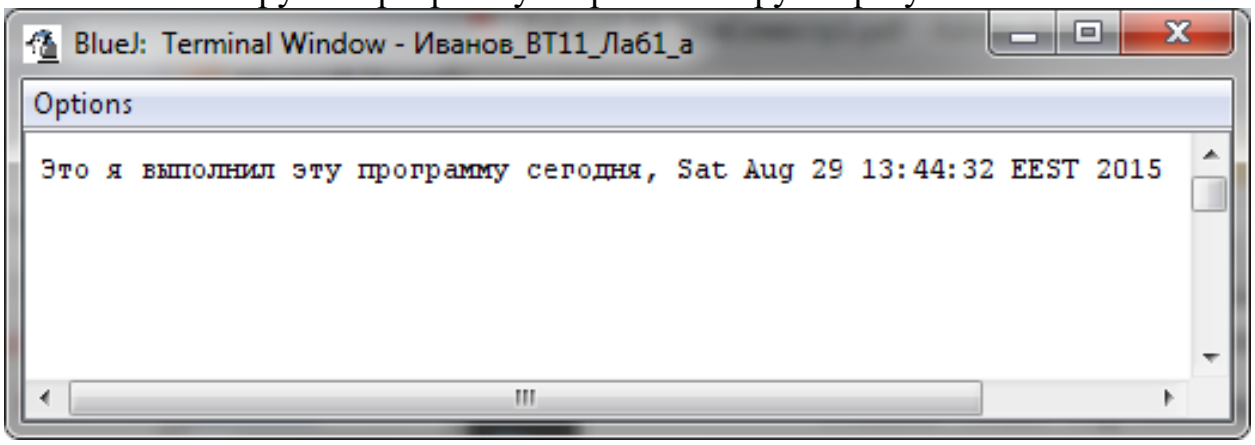

Рисунок 2.5. Окно терминала. Результат выполнения программы

Часть 3. Выполнение индивидуального задания.

Студент составляет программу (имя проекта: «Фамилия_Группа_Lab1b») согласно варианту задания (таблица 2.1) и оценивает результаты ее выполнения при помощи инструментальной системы BlueJ.

2.6. Варианты заданий

В качестве индивидуального задания на лабораторную работу предлагается разработать программу, выполняющую заданную операцию над операндами целого типа (int). В программе должны быть определены соответствующие переменные для хранения операндов и результата.
Программа должна осуществлять следующий вывод:

Программу выполнил:
Фамилия, имя, отчество студента,
Шифр группы,
Дата,
Вариант номер.
Название операнда1: значение операнда1,
Название операнда2: значение операнда2,
Название операции: значение результата.
Проверил:
Фамилия, имя, отчество преподавателя.

Проверьте работу программы на нескольких тестовых примерах.
Варианты заданий представлены в таблице 2.1.

Таблица 2.1 – Варианты заданий

Номер варианта	Операнд 1	Операнд 2	Операция
1	Число студентов в группе	Число групп в потоке	Число студентов в потоке
2	Число недель для выполнения проекта	Число дней в неделе	Число дней для выполнения проекта
3	Число дней в неделе	Число выходных дней	Число рабочих дней
4	Число мест в кинотеатре	Число залов	Число мест в зале
5	Число книг в учебном абонементе	Число книг в читальном зале	Число книг в библиотеке
6	Число тетрадей	Число учебников	Общее число товаров в накладной
7	Число домов	Число квартир в доме	Число семей, получивших квартиры.
8	Число абитуриентов, подавших заявления на специальность	Число бюджетных мест	Конкурс – человек на место
9	Число студентов	Число преподавателей	Среднее число студентов на одного преподавателя
10	Стоимость билета	Число мест в автобусе	Выручка за рейс
11	Себестоимость товара	Наценка	Цена товара
12	Длина бассейна	Ширина бассейна	Площадь бассейна
13	Объем помещения	Высота потолка	Площадь помещения
14	Численность управляющего персонала	Численность рабочих	Всего сотрудников на фабрике
15	Фонд заработной платы предприятия	Число сотрудников предприятия	Средняя зарплата
16	Вес товара	Вес тары	Общий вес
17	Длительность первой серии фильма в минутах	Длительность второй серии фильма в минутах	Длительность фильма в часах

Продолжение таблицы 2.1

Номер варианта	Операнд 1	Операнд 2	Операция
18	Число недель в семестре	Часов лекций по дисциплине в неделю	Всего часов лекций в семестре
19	Число студентов очной формы обучения	Число студентов заочной формы обучения	Всего студентов в университете.
20	Оклад	Отчисления	Зарплата на руки
21	Длина рулона ткани в метрах	Расход ткани на костюм в метрах	Число костюмов
22	Длительность серии фильма в минутах	Число серий	Длительность фильма в часах
23	Количество студентов, сдавших сессию на 4 и 5	Размер стипендии в рублях	Размер стипендиального фонда в рублях
24	Площадь стен помещения	Площадь рулона обоев	Число рулонов
25	Общая площадь участка	Площадь основания дома	Площадь приусадебного участка

2.7. Рекомендации по составлению отчета по лабораторной работе

Помните, что содержание отчета – важнейшее подспорье при защите лабораторной работы.

Отчет должен содержать все разделы, перечисленные в пункте 1 данных методических указаний, и соответствовать требованиям, изложенным в пункте 2.8.

2.8. Требования к документу отчета

Текстовая часть отчета по лабораторной работе выполняется с использованием печатающих устройств на одной стороне листа белой бумаги формата А4 с параметрами:

1) параметры страницы:
   поля, не менее:
   верхнее − 20 мм; левое − 20 мм;
   нижнее − 15 мм; правое − 15 мм;
2) параметры шрифта: гарнитура − TimesNewRoman, кегль (размер в пунктах) − 14; начертание − обычный; цвет − черный;
3) параметры абзаца:

выравнивание по ширине;
уровень – основной текст;
отступ слева – 0;
отступ справа – 0;
интервал перед – 0;
интервал после – 6;
первая строка – отступ, значение – 1 см;
междустрочный интервал − одинарный;

4) автоматическая расстановка переносов в основном тексте (исключая титульный лист и заголовки пунктов, подпунктов, рисунков и таблиц).

Правила оформления технических документов изложены в [2]. Разделы с одинарной нумерацией выполняются прописными буквами. В отчете требуется выделить эти разделы полужирным шрифтом и применить к ним выравнивание по центру. В технических отчетах такие пункты располагаются с новой страницы. В отчете по лабораторной работе допускается расположение пункта на той же странице. При этом до и после пункта пропускается одна строка. Точка в конце названия пункта не ставится.

Пункты с двойной и тройной нумерацией выполняют строчными буквами (первая – прописная), начинают с абзаца, до и после названия пункта пропускают одну строку, точка в конце названия не ставится. Можно выделить эти пункты полужирным шрифтом.

Данные варианта задания в пункте «Постановка задачи» приводятся в виде таблицы MicrosoftWord. Название таблицы (Таблица *номер* – Название) приводится над таблицей и выравнивается по левому краю таблицы. Нумерация таблиц осуществляется в рамках пункта с одинарной нумерацией, как это сделано в данных методических указаниях (Таблица 2.1 – Варианты заданий). Либо можно использовать сквозную нумерацию таблиц в рамках всего документа. Точка в конце названия таблицы не ставится. На таблицу обязательно нужно сделать ссылку в тексте отчета (например, «Варианты заданий приведены в таблице 2.1.»). При этом соответствующая таблица должна быть помещена на той же или на следующей странице. Если вся таблица не помещается на текущей странице, то часть таблицы (дублируя заголовок) можно перенести на следующую страницу. Вместо названия таблицы перед следующей частью таблицы приводится фраза «Продолжение таблицы *номер*» (например, Продолжение таблицы 2.1). Выравнивается фраза по левому краю таблицы.

Скриншот окна терминала с результатами выполнения программы необходимо оформить в виде рисунка. Название рисунка (Рисунок *номер* – Название) приводится под рисунком (подрисуночная подпись). Точка в конце названия не ставится. Нумерация рисунков производится подобно нумерации таблиц. Подрисуночная подпись выравнивается по центру рисунка. На рисунок, как и на таблицу, обязательно должна быть ссылка в тексте отчета (например, «Результаты выполнения программы изображены на рисунке 2.1.»).

Сокращения слова «рисунок» ни в подрисуночной подписи, ни в ссылке на рисунок не допускаются. Рисунок должен быть приведен на той же странице, где находится ссылка на него, или на следующей.

Рисунок, как и таблицу, можно расположить на нескольких страницах. Тогда на первой странице в качестве подрисуночной подписи приводится название рисунка (Рисунок *номер* – Название), а на остальных страницах – фраза продолжения (Продолжение рисунка *номер*).

На титульном листе часть текста, определяющую студента, выполнившего отчет, и преподавателя, проверившего отчет, оформить в виде отдельной надписи (с помощью инструмента «Надпись»).

Текст программы в соответствующем разделе отформатировать в соответствии со следующими параметрами:

1) шрифт: arial, 12 пт, полужирный;
2) абзац:
   - выравнивание по левому краю;
   - уровень – основной текст;
   - отступ слева – 0;
   - отступ справа – 0;
   - интервал перед – 0;
   - интервал после – 0;
   - первая строка – нет (отступа или выступа),
   - междустрочный интервал – множитель, значение – 1,2;
3) комментарии (обязательно должны присутствовать в тексте программы) выделить курсивом и синим цветом.

2.9. Контрольные вопросы

1) Опишите процесс создания проекта в инструментальной системе BlueJ.
2) Какие файлы входят в проект и каково их назначение?
3) Опишите процесс от создания до выполнения программы. Какая из трех моделей трансляции кода программы (компилятор, интерпретатор, компилятор+интерпретатор) используется в Java?
4) В чем преимущество используемой в Java системы трансляции-выполнения программ?
5) Опишите назначение текстового редактора. Файлы с каким расширением он создает?
6) Опишите назначение компилятора. Файлы с каким расширением он создает?
7) Что делает виртуальная Java-машина?
8) Что из себя представляет программа с линейной структурой на языке Java? Для чего предназначен метод main?
9) Что делает встроенный метод System.out.println( );
10) Как получить значение текущей даты?
11) Для чего используются идентификаторы? Назовите требования к ним.

12) Как определить переменную целого типа в программе?

13) Для чего использован оператор присваивания в программе.

14) Что такое переменная? Что такое выражение? Какие операции целочисленной арифметики выполнялись при вычислении выражения?

3. ЛАБОРАТОРНАЯ РАБОТА № 2. ОБРАБОТКА ДАННЫХ ПРОСТЫХ ТИПОВ. РАБОТА С ПАНЕЛЬЮ КОДА BLUEJ. ФОРМАТИРОВАННЫЙ ВЫВОД

3.1. Цель работы

Ознакомиться с простыми типами данных Java, научиться объявлять переменные и литералы этих типов и выполнять операции над ними, научиться применять оператор присваивания для данных простых типов, научиться применять метод System.out.printf() – метод форматированного вывода − для вывода на экран значений различных типов, научиться применять окно кода в BlueJ.

3.2. Постановка задачи

1) Ввести заданные операторы в окно кода BlueJ и проанализировать полученные результаты.
2) Разработать программу, в которой используется метод System.out.printf() для вывода в окно терминала данных, предусмотренных вариантом задания.

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.48-50, 52-64).

3.4. Краткие теоретические сведения

3.4.1. Простые типы данных

Простые, или примитивные, типы данных - это встроенные в язык Java типы, аналогичные типам данных большинства языков программирования. Эти типы могут применяться самостоятельно или используются как составные части более сложных типов данных, которые создают программисты.

Java позволяет использовать 8 простых типов данных:

byte, short, int, long –целочисленные,
float, double –вещественные обычной и двойной точности,
boolean – булевские (логические, представляющие истину или ложь),
char – символьные, представляющие символы Unicode.

Диапазоны значений для простых типов данных приведены на рисунке 3.1.

Тип	Содержит	Значение по умолчанию	Ширина (биты)	Min и Max значения
boolean	true false	false	1	Не применимо
char	U-символ	\u0000	16	\u0000 до \uFFFF
byte	Целое	0	8	-128..127
short	Целое	0	16	-32 768..32 767
int	Целое	0	32	-2 147 483 648 .. 2 147 483 647
long	Целое	0	64	От -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
float	IEEE754 плав.тчк.	0.0	32	От +/-3.40282347E+38 до +/-1.40239846E-45
double	IEEE754 плав.тчк.	0.0	64	От +/-1.79769313486231570E+308 до +/-4.94065645841246544E-324

Рисунок 3.1 – диапазоны значений для примитивных типов данных

3.4.2. Базовые понятия программирования

Вспомним некоторые понятия программирования, упомянутые в предыдущей лабораторной работе.

Переменные и именованные константы.

Переменные – элементы, значение которых при выполнении программы можно изменить, но тип изменить нельзя.

Переменные имеют имена и значения. Имя переменной – это название области памяти, в которой хранится значение переменной. Чтобы ввести переменную в употребление, ее необходимо объявить одним из двух способов:

1) тип имя;

2) тип имя = начальноеЗначение; (точка с запятой – это часть объявления!)

Объявление переменной – это приказ выделить для нее память.

Имена переменных пишут в стиле ―camelStyle, начиная с маленькой буквы, а каждое следующее слово в имени – с большой буквы.

Именованные константы – это переменные, значение которых определяется «при рождении», после чего изменять их значения запрещено.

Форма задания константы с именем: **final тип ИМЯ = значение;**

Слово final указывает, что значение нельзя изменить. Принято имена констант записывать БОЛЬШИМИ БУКВАМИ.

Оператор присваивания

Оператор присваивания заменяет старое значение некоторой переменной новым. После замены старое значение теряется, т.е. использовать его нельзя.

Форма оператора: переменная = выражение;

Переменная является приемником данных, а выражение – источником. Выражение определяет новое значение переменной. Равенство левой и правой частей наступает только после выполнения оператора присваивания и имеет смысл равенства значений источника и приемника. Порядок выполнения оператора: сначала, без учета левой части, вычисляется выражение с текущими значениями переменных. Затем полученное значение выражения становится значением переменной-приемника. Если в выражении используется та же переменная, скажем, x, что и слева, при вычислении выражения применяется имеющееся значение x, которое заменяется новым значением – значением выражения – после вычисления выражения. Например, пусть x=5.0. При выполнении оператора **x=x+1.2;** берется значение переменной **x**, равное 5, к нему добавляется число 1.2, и результат становится значением переменной **x**.

Язык Java требует, чтобы размер приемника был не меньше, чем размер результата. Когда это требование нарушается, возникает ошибка «Возможна потеря точности – possible loss of precision». Пример, при котором возникает такая ошибка: float x; x = 1.0; (размер слева – 32 бита, размер справа – 64 бита, т.к. 1.0 считается константой (литерал) типа double).

3.4.3. Использование окна кода BlueJ

Система BlueJ имеет специальное окно – панель (окно) кода (или окно команд) (рисунки 3.2, 3.3). Окно кода позволяет проверить фрагменты программы до их включения в проект. Действия в окне кода выполняются по строкам после нажатия Enter. Объявленные переменные запоминаются, т.е. повторно объявить переменную нельзя. Если окно кода не видно после запуска BlueJ, откройте проект, созданный ранее, и нажмите Виды – Показать панель кода. При необходимости вводимый в окно кода текст можно расположить в нескольких строках, при этом промежуточные строки заканчиваются нажатием Shift-Enter. Проверенные операторы можно вставить в текст программы, пользуясь копированием (Ctrl-C) и вставкой (Ctrl-V).

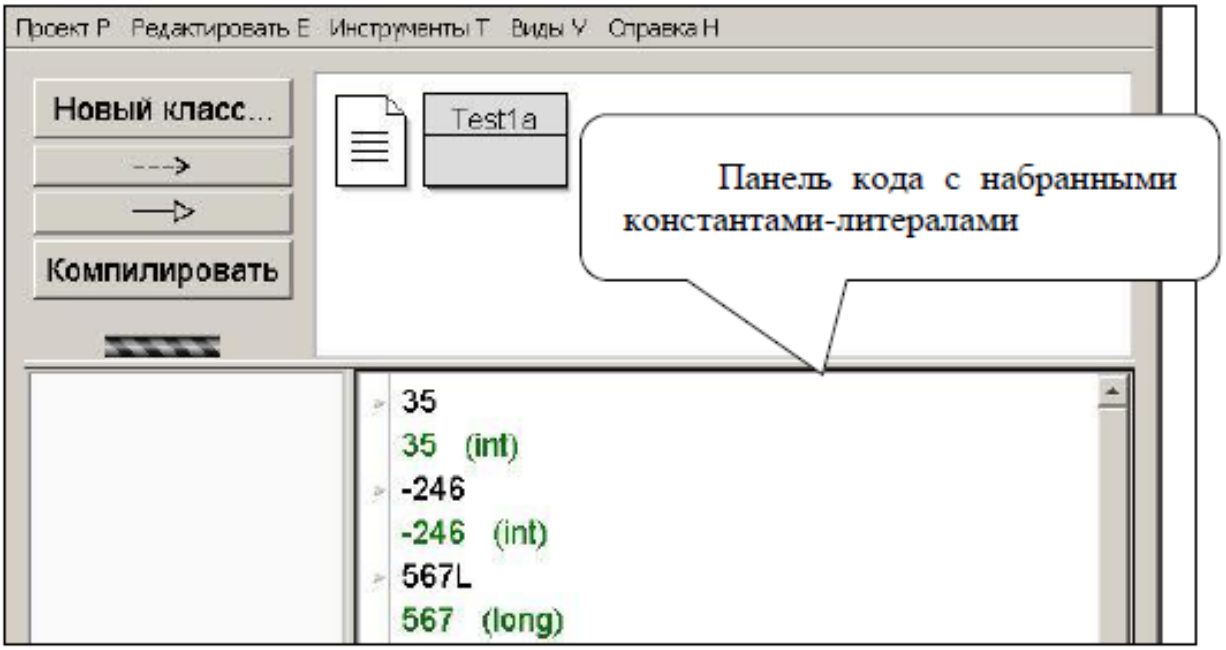

Рисунок 3.2. Окно кода. Примеры набора констант

Рисунок 3.3. Окно кода. Примеры набора выражений

3.4.4. Метод форматированного вывода System.out.printf()

В предыдущей работе использовался метод System.out.println(), который выводил в окно терминала символьную строку (объект типа String).

В этой работе рекомендуется применить метод форматированного вывода System.out.printf(), который более удобен, например, для представления результатов в виде аккуратной таблицы. Вызов метода имеет форму:

System.out.printf (format, list);

Строка форматов **format** содержит **форматные коды**, по одному на каждый элемент списка выводимых переменных **list**. Форматный код имеет представленную ниже структуру (в скобки [и] заключены необязательные элементы, т.е. в программе такой элемент или присутствует без скобок, или отсутствует)

%[номер_аргумента$][флаги][место][.дробь]тип

Флаги **- + 0 , (** определяют дополнительные характеристики выводимых данных: минус – выравнивание влево, + – обязательный вывод знака числа, пробел – пробел вместо плюса для положительных чисел, ноль – заполнение старших пустых позиций числа нулями, запятая – локализованный разделитель в числах, скобка – отрицательные числа заключаются в скобки.

Параметр тип определяет вариант преобразования внутреннего представления данных во внешнюю форму:

- s – String (строка символов);
- c – char (символ);
- d – десятичное целое;
- o – восьмеричное целое;
- x – шестнадцатеричное целое;
- f – действительное в фиксированном формате (в форме 999.9999);
- e – действительное в математическом формате (в форме 0.999E99);
- tD – дата в форме «месяц-день-год»;
- tF – дата в формате «год-месяц-день»;
- tT – время в формате «часы:минуты:секунды».

Параметры место и дробь определяют соответственно количество позиций для выводимых данных (ширину поля) и количество цифр в дробной части (применяются с параметрами типа d и f). Например, %08.3f применяется для вывода действительных данных. Под данные отводится 8 позиций (с учетом знака и разделяющей точки). Из них под дробную часть отводится 3 позиции. При необходимости в целую часть добавляются ведущие нули, чтобы полностью использовать 8 позиций.

Элементы между форматными кодами выводятся как литералы, в частности, \t – табуляция, \n – переход на новую строку (**%n** также означает переход на другую строку).

Примеры:

```
printf("Hello %s!", "World"); //  "Hello World!"
printf("%7d", 1); //  "      1" – минимум 7 позиций
printf("%07d", 1); //  "0000001" – начальные позиции
                                  //заполнены нулями
printf("%.10f", Math.PI);  //  "3,1415926536" – с точностью
                           // до 10 знаков после запятой
printf("%tF", new Date()); //  "2011-01-27"
printf("%tT", new Date()); //  "22:42:37"
```

Фрагмент программы:

```
int a = 256;
double b = 312.678951236;
String str = "форматированный вывод";
System.out.printf("Демонстрируем %s: a=%08d(10)=%x(16), b=%010.4f\n",
                  str,a,a,b);
```

Данный фрагмент повлечет за собой следующий вывод в окно терминала:

Демонстрируем форматированный вывод: a=00000256(10)=100(16), b=00312,6790

Элементы между форматными кодами (выделены красным в вызове метода printf()) выводятся как строковые литералы (константы).

3.5. Порядок выполнения работы

3.5.1. Работа в окне кода BlueJ

1) Введите целые константы в окне кода, нажимая после каждой константы Enter: **`35 -246 5671L.`**

2) Введите выражения **`4*3+5 4+3*5 25-3/5-6+20/3 5%6`** («пять по модулю шесть» − остаток от деления 5 на 6) **`7%6 7%5.`** После ввода каждого выражения нажимайте Enter. Обратите внимание на тип и результат.

3) Наберите выражения **`1==1 1<5 2<=5 2>6 3/2>1 2!=9.`** После ввода каждого выражения нажимайте Enter. В этих выражениях используются операции сравнения. Результат операции (и выражения) имеет логический тип (boolean). Если выражение верно, то его значение равно true, иначе – false. Проверяйте результат каждый раз после нажатия Enter.

4) Введите в панели кода действительные литералы, каждый в отдельной строке **`3.1415 2.71823F 0.314E+1.`** Проанализируйте результаты (что происходит после нажатия клавиши Enter).

5). Наберите в панели кода выражения **`25-3.5-6+21.3 5/6.0 5.0/6 3.7f/5.3f`**. Не забывайте нажимать Enter! Обратите внимание на значение результата и на его тип.

6). Наберите в панели кода выражения **1==1.0 1.0<1.5 2.3<=2.3f 1.5==1.5f**

7) Наберите в панели кода: **`double x; float a=3.5; float d=2.5;`** (После точки с запятой нажимайте Enter). Проанализируйте сообщения системы.

8) Наберите в панели кода: **`x=3.5;`** (тип x уже задан) **`float a=3.5f; double d=2.5f;`** (После точки с запятой нажимайте Enter). Проверьте значения переменных, введя имя и Enter.

9) Наберите **`d=d+3.75; x=a+7.5f;`** Проверьте значения переменных.

10) Введите **`final float K=5.4f;`** Проверьте значение K. Наберите оператор **`K=5.45;`** . Обратите внимание на сообщение. Наберите **`final double PI=3.1415926;`** , вычислите длину окружности с радиусом 10: **`2*PI*10`**.

3.5.2. Разработка программы с использованием метода форматированного вывода System.out.printf()

Разработайте программу согласно варианту задания (таблица 3.1), проведите ее отладку и испытание на нескольких тестовых примерах.

3.6. Варианты заданий

В качестве индивидуального задания на лабораторную работу предлагается разработать программу, которая выполняет следующие действия:
1) определяет и инициализирует (задает тип и значение) переменные str, a, b, c, d;
2) осуществляет следующий вывод:

Привет, значение_str: a=значение_a, b=значение_b, c=значение_c, d=значение_d!

Значения переменных должны выводиться в формате, предусмотренном вариантом задания. После вывода строки, должен быть осуществлен переход на следующую строку.

После выявления синтаксических ошибок программа должна быть откомпилирована и запущена не менее трех раз (с различными значениями переменных). В качестве тестовых значений переменных студент должен подобрать такие значения, которые проверяют все возможности программы. Например, проверку обязательного вывода знака числа нужно осуществить на положительных и отрицательных значениях. Желательно также проверить, как будет вести себя ваша программа, если заданное значение переменной не помещается в формат, заданный в методе printf().

Варианты индивидуальных заданий представлены в таблице 3.1.

Использованы следующие условные обозначения:
С10 – десятичная система счисления;
С16 – шестнадцатеричная система счисления;
С8 – восьмеричная система счисления;
Ш – ширина (минимальное количество цифр);
Т – точность (число цифр после запятой);
НН – наличие ведущих нулей;
ОН – отсутствие ведущих нулей;
ВЗ – обязательный вывод знака.

Таблица 3.1 – Варианты заданий

Номер варианта	Строка str (String)	Целое a (int)	Целое b (short)	Целое c (byte)	Действительное d (double)	Действительное f (float)
1	Фамилия_Группа:	С10, Ш8, НН, ВЗ	С8	С16	Ш10, Т5, НН,	Ш8, Т3, ОН
2	Фамилия_Группа:	С16	С10, Ш5, ОН	С8	Ш5, Т3, ОН, ВЗ	Ш7, Т2, НН
3	Фамилия_Группа:	С8	С16	С10, Ш6, ОН	Ш8, Т4, НН	Ш6, Т3, ОН, ВЗ
4	Фамилия_Группа:	С16	С8	С10, Ш8, НН	Ш7, Т3, ОН, ВЗ	Ш9, Т4, НН
5	Фамилия_Группа:	С10, Ш5, ОН, ВЗ	С8	С16	Ш8, Т2, НН	Ш10, Т4, ОН
6	Фамилия_Группа:	С10, Ш5, ОН	С8	С16	Ш8, Т2, ОН, ВЗ	Ш6, Т3, НН
7	Фамилия_Группа:	С8	С10, Ш9, НН	С16	Ш10, Т4, ОН	Ш8, Т2, НН, ВЗ
8	Фамилия_Группа:	С16	С10, Ш7, ОН, ВЗ	С8	Ш12, Т6, НН	Ш6, Т3, ОН
9	Фамилия_Группа:	С16	С8	С10, Ш8, НН,	Ш11, Т5, ОН, ВЗ	Ш9, Т4, НН
10	Фамилия_Группа:	С10, Ш7, ОН, ВЗ	С8	С16	Ш14, Т5, НН,	Ш8, Т3, ОН
11	Фамилия_Группа:	С10, Ш8, НН, ВЗ	С8	С16	Ш10, Т5, НН,	Ш8, Т3, ОН
12	Фамилия_Группа:	С16	С10, Ш5, ОН	С8	Ш5, Т3, ОН, ВЗ	Ш7, Т2, НН
13	Фамилия_Группа:	С8	С16	С10, Ш6, ОН	Ш8, Т4, НН	Ш6, Т3, ОН, ВЗ
14	Фамилия_Группа:	С16	С8	С10, Ш8, НН	Ш7, Т3, ОН, ВЗ	Ш9, Т4, НН
15	Фамилия_Группа:	С10, Ш5, ОН, ВЗ	С8	С16	Ш8, Т2, НН	Ш10, Т4, ОН
16	Фамилия_Группа:	С10, Ш5, ОН	С8	С16	Ш8, Т2, ОН, ВЗ	Ш6, Т3, НН
17	Фамилия_Группа:	С8	С10, Ш9, НН	С16	Ш10, Т4, ОН	Ш8, Т2, НН, ВЗ
18	Фамилия_Группа:	С16	С10, Ш7, ОН, ВЗ	С8	Ш12, Т6, НН	Ш6, Т3, ОН

Продолжение таблицы 3.1

Номер варианта	Строка str	Целое a (int)	Целое b (short)	Целое c (byte)	Действительное d (double)	Действительное f (float)
19	Фамилия_Группа:	С16	С8	С10, Ш8, НН,	Ш11, Т5, ОН, ВЗ	Ш9, Т4, НН
20	Фамилия_Группа:	С10, Ш7, ОН, ВЗ	С8	С16	Ш14, Т5, НН,	Ш8, Т3, ОН
21	Фамилия_Группа:	С10, Ш8, НН, ВЗ	С8	С16	Ш10, Т5, НН,	Ш8, Т3, ОН
22	Фамилия_Группа:	С16	С10, Ш5, ОН	С8	Ш5, Т3, ОН, ВЗ	Ш7, Т2, НН
23	Фамилия_Группа:	С8	С16	С10, Ш6, ОН	Ш8, Т4, НН	Ш6, Т3, ОН, ВЗ
24	Фамилия_Группа:	С16	С8	С10, Ш8, НН	Ш7, Т3, ОН, ВЗ	Ш9, Т4, НН
25	Фамилия_Группа:	С10, Ш5, ОН, ВЗ	С8	С16	Ш8, Т2, НН	Ш10, Т4, ОН

3.7. Рекомендации по составлению отчета по лабораторной работе

Отчет должен соответствовать требованиям, изложенным в пункте 2.8, и содержать разделы, перечисленные в пункте 1 данных методических указаний.

3.9. Контрольные вопросы

1) Какие простые типы данных существуют в Java?
2) От чего зависит диапазон значений числовых типов?
3) Какой из типов обеспечивает б*о*льшую точность double или float?
4) Что такое литерал?
5) Как задается переменная?
6) Как задается именованная константа?
7) Как задать литерал типа float?
8) Как задать литерал типа long?
9) Опишите, как работает оператор присваивания?
10) Когда возникает ошибка «Потеря точности»?
11) Какие возможности предоставляет окно кода BlueJ?
12) Чему равно значение выражения 2<5? Какого оно типа?
13) Чему равно значение выражения 10<=5? Какого оно типа?
14) Какие возможности предоставляет оператор форматированного вывода System.out.printf (format, list).

4. ЛАБОРАТОРНАЯ РАБОТА № 3. ОБРАБОТКА ЦЕЛОЧИСЛЕННЫХ ДАННЫХ.

4.1. Цель работы

Получить навыки объявления в программе переменных и литералов, применения арифметических операций для данных целых типов, применения оператора присваивания, закрепить знания по теме «Представление целых чисел в компьютере.

4.2. Постановка задачи

1) Ознакомиться с принципами хранения и обработки целочисленных данных в Java.

2) Разработать и отладить программу, демонстрирующую выполнение операций над данными целого типа.

4.3. Внеаудиторная подготовка

Для подготовки к лабораторной работе следует изучить краткие теоретические сведения, приведенные в пункте 4.4 методических указаний. Рекомендуется также ознакомиться с [1] (с.48-50, 52-64).

4.4. Краткие теоретические сведения

Характеристики целочисленных типов данных приведены в таблице 4.1.

Таблица 4.1 – Целочисленные типы данных и их характеристики

Тип	Значение	Значение по умолчанию	Размер (биты)	Min и Max значения	Класс-оболочка
byte	Целое	0	8	от -128 до 127	Byte
short	Целое	0	16	от -32768 до 32767	Short
int	Целое	0	32	от -2147483648 до 2147483647	Integer
long	Целое	0	64	-9223372036854775808 9223372036854775807	Long

Для каждого целочисленного типа существует библиотечный класс-оболочка [1, 3], содержащий полезные методы для работы с целочисленными данными. Названия этих классов приведены в четвертом столбце таблицы 4.1.

Внутреннее и внешнее представление целочисленных данных.

Внутренне представление – это то, которое «видит» процессор, внешнее – то, которое видит и применяет при записи литералов (в данном случае, целых чисел) программист. Внутреннее представление целочисленных данных – двоичное, количество битов приведено в столбце «Размер» таблицы 4.1. Под знак числа выделен левый бит. Значение 0 этого бита имеют положительные числа, значение 1 – отрицательные. Формат двоичного целого числа представлен на рисунке 4.2. Для типа byte n=8, для типа short n=16, для типа int n=32, для типа long n=64.

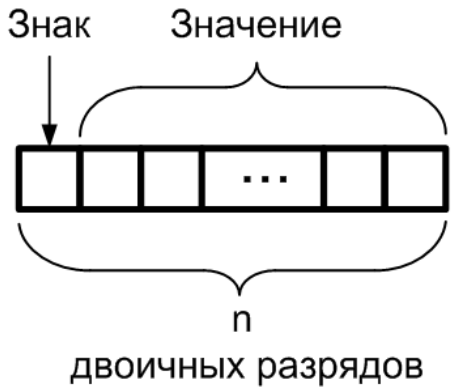

Рисунок 4.2 – Формат целого двоичного числа

Отрицательные числа представляются Java в дополнительных кодах следующим образом:

Знак=1;

Значение* = 2^{n-1} ─ |Значение|,

где Значение* - *значение числа в дополнительном коде,*

Значение – *обычное представление двоичного числа.*

Принято, что для любого положительного числа дополнительный код совпадает с прямым кодом (обычным представлением числа в двоичной позиционной системе счисления).

Все операции с целыми числами выполняются в двоичной системе счисления. Использование дополнительного кода для представления двоичных чисел позволяет процессору корректно выполнять операции сложения и вычитания на двоичном сумматоре.

Литералы (целые числа, присутствующие в программе) автоматически переводятся из внешнего представления (последовательность символов) во внутреннее (последовательность битов двоичного целого числа). Метод println() автоматически преобразует числа из внутреннего во внешнее представление. Все другие преобразования выполняются под контролем программиста.

Запись целочисленных литералов в Java-программах.

Восьмеричные значения обозначаются в Java ведущим нулем.

Десятичные значения не могут иметь ведущего нуля.

Значение 09, встреченное в программе, вызовет ошибку компилятора, т.к. цифра 9 ─ вне восьмеричного диапазона от 0 до 7.

Шестнадцатеричную константу обозначают с ведущими нулями: 0x или 0X.

Целые литералы создают значение типа int (32-разрядное целое число).

Когда литеральное значение назначается byte- или short-переменной, ошибка не генерируется, если это значение находится в пределах диапазона целевого типа.

Целый литерал может всегда назначаться переменной типа long. Однако, чтобы задать длинный литерал, нужно явно сообщить компилятору, что значение имеет тип long. Это осуществляется добавлением символа L (в верхнем или нижнем регистре).

Например, 0x7FFFFFFFFFFFFFFL или 92233720368547758O7L – самый большой целочисленный литерал.

Операции для целочисленных типов данных.

Арифметические операции – (+ - * / %) – (сложение, вычитание умножение, деление нацело, вычисление остатка от деления). Результат этих операций – целочисленный.

Операции сравнения (или отношения) – (< > <= >= == !=) – (меньше, больше, меньше или равно (не больше), больше или равно (не меньше), равно, не равно). Результат сравнения – булевское значение **true** (истина) или **false** (ложь), например, 2<3 имеет значение true, а 2>3 – значение false.

Поразрядные логические операции будут рассмотрены в следующей лабораторной работе.

Арифметические операции с целочисленными данными выполняются точно, без округления. При выполнении целочисленного деления, дробная часть результата отбрасывается. В

Виду ограниченного диапазона (таблица 4.1), сложение и вычитание могут дать «переполнение» – результат, выходящий за границы диапазона. Переполнение выглядит как отрицательный результат при положительных слагаемых или как положительный результат при отрицательных слагаемых. Например, если к максимальному положительному int, равному 0x7FFF FFFF, прибавить 1, получится наименьшее целое отрицательное число 0x8000 0000. Если это число сложить с максимальным положительным (0x8000 0000 + 0x7FFF FFFF), получится число 0xFFFF FFFF (двоичное представление числа -1 в дополнительном коде), которое на единицу меньше нуля (легко проверить, прибавив к нему 1).

При соединении ряда операций в одном выражении следует учитывать приоритеты операций. Так, арифметические операции имеют больший приоритет, чем операции сдвигов и поразрядные операции, а операции над одним операндом имеют больший приоритет, чем операции над двумя операндами.

При вычислении значения выражения Java автоматически расширяет (повышает) тип каждого byte- или short-операнда и результата до int (т.к.

результат сложения двух коротких операндов с одинаковыми знаками может не поместиться в короткий формат).

Это может вызывать плохо распознаваемые ошибки во время компиляции.

Например, казалось бы, благополучный код

```
byte b=50;
b=b*2;
```

вызовет ошибку компиляции «Возможна потеря точности – possible loss of precision», т.к. значение выражения b*2 имеет тип int, а целевая переменная b имеет тип byte. Такая же ошибка возникнет и при компиляции строки кода **int i = 2L;** (переменной типа int присваивается значение типа long).

Вообще, когда размер переменной слева меньше, чем размер результата справа, возникает ошибка «Возможна потеря точности – possible loss of precision».

Для преодоления этой трудности есть два пути:

1) числа длиной 1 байт хранить, например, в переменных типа int;
2) использовать явное преобразование типа значения в правой части оператора присваивания:

```
byte b=50;
b=(byte)(b*2);  .
```

Следует отметить, что тип int – наиболее универсален и эффективен и должен быть использован для расчетов, индексации элементов массива или выполнения целочисленных операций.

Может показаться, что использование short или byte экономит память, но нет никакой гарантии, что Java не будет внутренне так или иначе расширять эти типы до int, Помните, что тип определяет поведение, а не размер. Единственное исключение – массивы, где тип byte гарантирует использование только одного байта на элемент массива, тип short будет забирать 2 байта, а int 4 байта [1].

4.5. Порядок выполнения работы

1) Разработайте программу согласно варианту задания (таблица 4.2),
2) Обоснуйте выбор типа целочисленных переменных.
3) Проведите отладку программы и испытание на нескольких тестовых примерах.

4.6. Варианты заданий

В качестве индивидуального задания на лабораторную работу предлагается разработать программу, демонстрирующую выполнение арифметических операций над целыми числами.

В программе задать целочисленные переменные a, b, c. Вывести на экран в двоичной, шестнадцатеричной, восьмеричной и десятичной системах счисления

значения a, b, c, а также результаты выполнения операций. -a, a+b, a-b, a*b, a/b, a%b, a++, b--. Например,

a+b=100000000(2)=100(16)= 400(8)=256 (10); .

Подобный вывод можно осуществить одним вызовом метода printf(). Для получения представления двоичного числа в виде строки (String) используйте библиотечную функцию **Integer.toBinaryString(x),** где x – аргумент типа int.

Сравнить с результатами ручных расчетов.

Варианты заданий представлены в таблице 4.2.

Таблица 4.2 – Варианты заданий

Номер варианта	a	b	c
1	356	725	(b-a)%a+56
2	125	49	(a+b)/b-a+100%b
3	1024	541	a%b*2-b/4
4	986	25	a/(b+20)-253%b
5	1255	345	(a+b)/(b+20)%20
6	236	16	a/b%5+12*b
7	512	48	a/b+(a%b)*2
8	2056	185	(a+2*b)/(a-5*b)%9
9	145	312	a/16+b%16-10*a
10	1345	1200	(a-b)/5+(b-500)%5*3
11	712	256	(b-a+a/4+b*4)%10
12	123	56	a*b-(a+b/2)%7
13	875	298	(a%b-a/b)*20
14	1421	1200	(a-b/2+b%10)*3
15	127	512	a/b*(a-b)%15+25
16	2024	341	-(a/b+a%b)*5
17	456	1348	(b/a+b%a*3)/10
18	25	354	(a*a-b)%25/2
19	849	315	((a-b)%b-a/b)*3
20	5	7	(a*a+b*b*b)/10%5
21	15	12	((a*a-b*b)+1024)%12/3
22	1500	2800	((b-a)/10)*((b-a)/10)%20
23	456	1234	(2*a+b/45)*3%14
24	1155	956	((b-a)*2560%b)/12
25	1024	2048	-2*(a%100+b%200)/5

4.

7. Рекомендации по составлению отчета по лабораторной работе

Отчет должен соответствовать требованиям, изложенным в пункте 2.8, и содержать разделы, перечисленные в пункте 1 данных методических указаний.

4.9. Контрольные вопросы

1) Что такое литерал?
2) Как в программе задать целочисленные литералы в различных системах счисления?
3) Как получить представление двоичного целого числа в виде строки символов (String)?
4) Какие арифметические операции и операции сравнения допустимы для целочисленных типов?
5) Как выполняется операция целочисленного деления?
6) Как выполняется операция, обозначаемая символом %?
7) Что такое переменная? Как объявляется переменная целочисленного типа?
8) Как задать именованную константу целочисленного типа?
9) Какие ошибки могут возникать при компиляции оператора присваивания, использующего переменные и выражения различных целых типов? Приведите примеры.
10) Какой тип будет у результата выражения, в котором выполняются операции над переменными типа short и byte?
11) Какой тип предпочтительно выбирать для одиночных (не массивов) целочисленных переменных и почему?
12) Внутреннее представление чисел:
0 0100111,
1 1111101,
1 1111111,
0 0000000 0100111,
1 1111111 1111101,
1 1111111 1111111,
0 0000000 00000000 00000001 000011100,
1 1111111 11111111 11111111 101011010.
Запишите эти числа в десятичной, восьмеричной и шестнадцатеричной системах счисления. Какого они типа?
13) Назовите максимальное значение (в шестнадцатеричной системе счисления), которое можно записать в переменную типа byte, short, int и long.
14) Чему равны значения выражений 10<=3, 10!=3, 5>3+1? Какого они типа?
15) Фрагмент кода: int a=5; int b=3; f=a<b+1; . Какого типа переменная f и чему равно ее значение?

5. ЛАБОРАТОРНАЯ РАБОТА № 4. ОБРАБОТКА ЧИСЛОВЫХ ДАННЫХ ВЕЩЕСТВЕННЫХ ТИПОВ

5.1. Цель работы

Закрепить навыки объявления переменных и констант, освоить операции для данных вещественных типов, закрепить навыки применения оператора присваивания, исследовать форматы внутреннего представления вещественных чисел.

5.2. Постановка задачи

1) Изучить принципы хранения и обработки числовых данных вещественных типов (типов с плавающей точкой) в Java. Провести исследования в окне кода BlueJ, описанные в п. 5.5.1 методических указаний.

2) Изучить пример выполнения индивидуального задани**я**, приведенный в п. 5.7 методических указаний. Разработать и отладить программу, демонстрирующую выполнение операций над данными вещественного типа.

3) Исследовать внутреннее представление данных вещественных типов в формате IEEE-754 для значений, являющихся результатами выполнения программы.

5.3. Внеаудиторная подготовка

Для подготовки к лабораторной работе следует изучить краткие теоретические сведения, приведенные в пункте 5.4 методических указаний. Рекомендуется также ознакомиться с [1] (с.56-58) и [2].

5.4. Краткие теоретические сведения

5.4.1. Вещественные числа и соответствующие им машинные форматы

В данной работе исследуются типы float и double (вещественные данные одинарной и двойной точности), которые применяются для выполнения инженерных и научных расчетов. Характеристики этих типов приведены в таблице 5.1.

Таблица 5.1 – Целочисленные типы данных и их характеристики

Тип	Значение	По умолчанию	Размер (биты)	Min и Max значения	Класс-оболочка
float	IEEE754 плав.тчк.	0.0	32	От +/-3.40282347E+38 до +/-1.40239846E-45	Float
double	IEEE754 плав.тчк.	0.0	64	От +/- 1.79769313486231570E+308 до +/-4.94065645841246544E-324	Double

Для каждого из вещественных типов существует библиотечный класс-оболочка [1, 3], содержащий полезные методы для работы с данными с плавающей точкой. Названия этих классов приведены в шестом столбце таблицы 4.1.

Если данные по смыслу задачи могут иметь дробную часть, применяют вещественные (синоним – действительные) данные. Java реализует стандартный (IEEE-754) набор типов с плавающей точкой и соответствующие операции с ними. Существует два вида типов с плавающей точкой float (4 байта) и double (8 байтов), которые представляют числа с одинарной и двойной точностью соответственно

Выбор между float и double делают, исходя из необходимой точности представления данных.

Представление данных с плавающей точкой в Java.

У вещественного числа X можно выделить три части: знак S, мантисса M и порядок P. Знак S числа X – это знак его мантиссы. Мантисса M представляет собой значащую часть (цифры) числа, порядок P указывает на сколько разрядов влево (отрицательный порядок) или вправо (положительный порядок) нужно перенести запятую в мантиссе M. Значение вещественного числа X можно определить по формуле

$$X = M * q^p,$$

где q – основание системы счисления, в которой представлено число X.

Вещественное число считается нормализованным, если его мантисса

$$\frac{1}{q} \le |M| < 1 \quad ,$$

Например, у десятичного числа $X_{10} = 327{,}253 = 0{,}327253 \times 10^3$
нормализованная мантисса $M = 0{,}327253.$
Соответственно порядок $P = 3$
Еще примеры:

$$X_{10} = 0{,}000327253 = 0{,}327253 \times 10^{-3};$$
$$M = 0{,}327253; \qquad P = -3.$$

$$X_{10} = 0{,}000327253 = 0{,}327253 \times 10^{-3};$$
$$M = 0{,}327253; \qquad P = -3.$$

$$X_2 = 101100{,}1011 = 0{,}1011001011 \times 2^6;$$
$$M = 0{,}1011001011\ ; \qquad P = 6_{10}.$$

$$X_2 = 0{,}00001011001011 = 0{,}1011001011 \times 2^{-4};$$
$$M = 0{,}1011001011\ ; \qquad P = -4_{10}.$$

Как мы знаем, числа представляются в ЭВМ в двоичной системе счисления. На рисунке 2.1 изображен стандартный формат представления чисел с плавающей точкой IEEE-754 [5], который используется в Java.

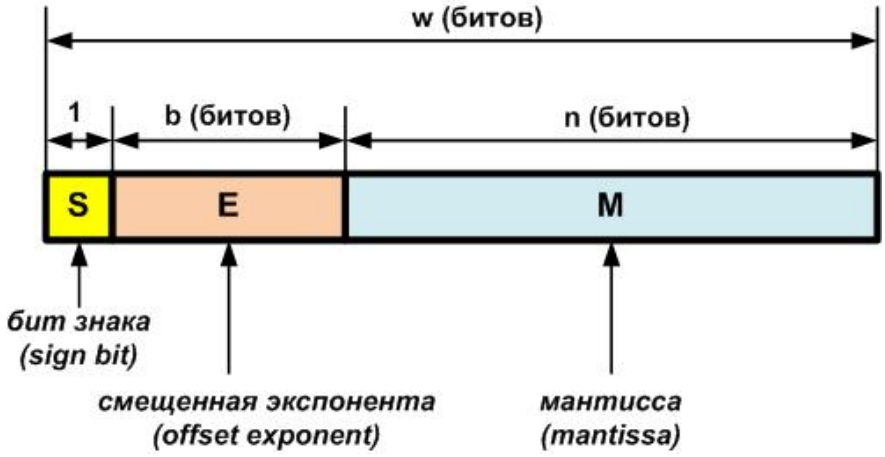

Рисунок 5.1 – Формат IEEE-754

S – бит знака числа: если S=0, число положительное,
если S=1, число отрицательное;

E – смещенный порядок (экспонента) двоичного числа;

$2^{(b-1)}-1$ — заданное смещение экспоненты (позволяет задавать только положительную экспоненту, т.е. не выделять дополнительный бит для ее знака);

$P = E - (2^{(b-1)}-1)$ — несмещенное (настоящее) значение порядка;

M — остаток мантиссы двоичного нормализованного числа с плавающей точкой.

Двоичное число с плавающей точкой создатели формата IEEE-754 считают нормализованным, если 1<=Мантисса<2. Такая мантисса всегда начинается с 1. Нет смысла хранить эту единицу в отведенных n битах. Поэтому в n битах хранят только дробную часть — остаток от мантиссы **M** (экономят один бит точности).

Формула вычисления десятичных чисел с плавающей точкой, из чисел, представленных в стандарте IEEE-754:

$$F = (-1)^S 2^{(E-2^{(b-1)}+1)}(1 + M/2^n)$$

Цветом сделан акцент на прибавление 1, с которой начинается мантисса любого нормализованного числа (по версии IEEE-754) , и которую, поэтому, нет смысла хранить, расходуя на нее один лишний бит.

Деление остатка от мантиссы **M** на 2^n обозначает отрицательные степени двойки, соответствующие весам разрядов **M**, например, $2^{-1}=1/2$

Интерпретации формата IEEE-754 для типа с одинарной точностью (float) и типа с двойной точностью (double) приведены на рисунке 2.2.

Заметим, что точность представления чисел определяется числом разрядов мантиссы. Число разрядов порядка определяет диапазон чисел (таблица 5.1).

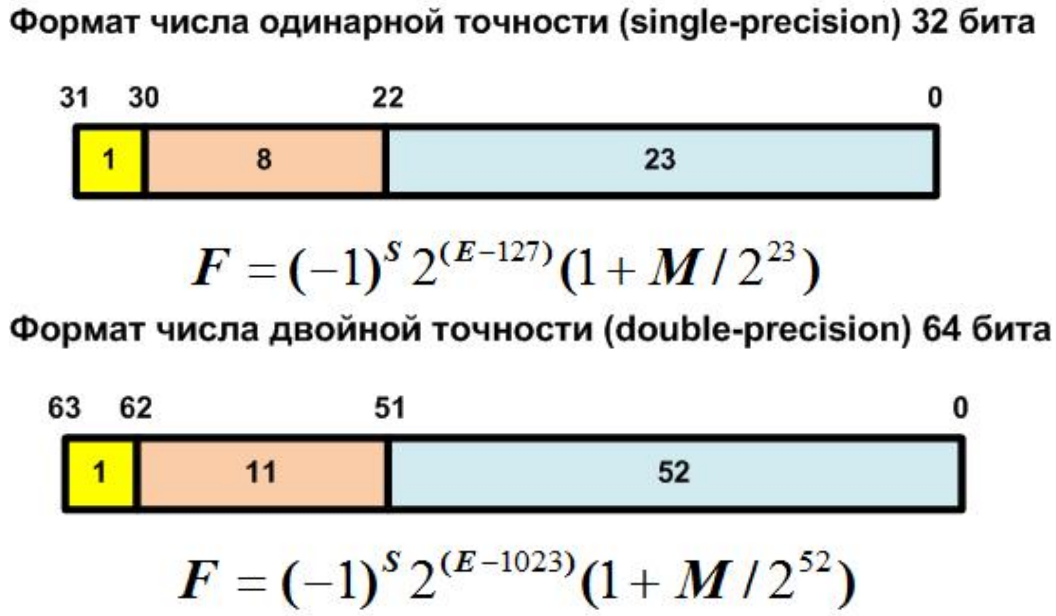

Рисунок 5.2 – Интерпретации формата IEEE-754 для типов с одинарной и двойной точностью

Пример представления вещественного числа в формате float приведен на рисунке 2.3.

Рисунок 5.3 – Пример представления вещественного числа в формате float

Операции над вещественными данными.

С вещественными данными можно выполнять обычные арифметические операции (+ - * /), результат которых – тоже вещественное число. Сравнение действительных данных выполняется операциями <, >, <=, >=, ==, != (меньше, больше, меньше или равно (не больше), больше или равно (не меньше), равно, не равно). Следует проявлять внимательность при сравнении действительных данных и действительных литералов (описаны ниже), поскольку действительные данные представляются с погрешностью.

Каждому простому типу в языке Java соответствует объектный тип – так называемый класс-оболочка (wrapper class). Классы-оболочки (для float и double это **Float** и **Double**) содержат полезные и интересные методы для проверок свойств данных или для их преобразования в другие типы. Например, метод

`static long doubleToLongBits(double value),`

определенный в классе Double, возвращает представление заданного значения value типа double в соответствии с форматом IEEE-754;

`static int floatToIntBits(float value),` определенный в классе Float возвращает представление заданного значения value типа float в соответствии с форматом IEEE-754.

Подробная информация о классах-оболочках имеется в документации по Java.

Константы – литералы действительных типов.

Как мы уже знаем, литералы – это данные, значение которых целиком определяется самой записью этих данных, т.е. является частью текста программы.

Литералы с плавающей точкой могут быть выражены в стандартной (с фиксированным числом знаков после десятичной точки) или научной (экспоненциальной) форме.

Стандартная форма состоит из целого компонента, за которым следует десятичная точка, а далее – дробная компонента, например, 2.0, 3.14159, 0.25.

Научная форма представления использует стандартные обозначения: число с плавающей точкой (мантисса) плюс суффикс, определяющий степень 10, на которую должно быть умножено число. Экспонента (возведение в степень) обозначается буквой E или e (английского алфавита). За ней следует положительное или отрицательное число – порядок степени. Например, ─ 6.022E23, 314159E ─05, 2e+100.

Научная форма вещественного числа проиллюстрирована на рисунке 2.4.

Рисунок 5.4 – Научная форма представления вещественного числа

В программе можем встретить, например, следующие строки:
double a = ─299.555E─2;
float b = 0.5f ;

По умолчанию литералы с плавающей точкой Java имеют double-точность. Чтобы определить float-литерал, следует добавить в конец его записи символ F или f. Можно также явно определить double-литерал, добавляя D или d в конец его записи, но такое определение будет избыточным.

Примеры:
для типа float: 3.1415F или 3.1415f (F или f обозначает тип float);
для типа double: 3.1415 (в фиксированном формате) или 0.314E+1 (в научном формате), 234.525d (явно указан тип).

Выражения.

Выражения позволяют комбинировать константы, переменные и операции для выполнения сложных вычислений.

Соединение в одном выражении нескольких операций требует учета их старшинства. Порядок выполнения операций в выражении:

1) выражение в круглых скобках,

2) умножение и деление,

3) сложение и вычитание.

Обычно выражения – это правые части формул для вычисления значений переменных. Если операцию требуется выполнить с данными разного размера (см. таблицу 2.1) или типа, то более простой тип автоматически преобразуется в более сложный (целочисленные преобразуются в действительные, меньшие по размеру преобразуются в большие по размеру). Например, float-данные автоматически преобразуется в double, если складываются с double-данными. А преобразование (из double во float) должно указываться программистом явно, т.к. связано с уменьшением размера, т.е. с потерей значащих цифр. Указание дается в форме (float)(выражение). Например, следующий фрагмент кода требует явного преобразования типа:

double b=2.5;

float c=5.25f;
float a= **(float) (b+c)**; .

Особенности выполнения операций с действительными данными.

Операции с действительными данными выполняются с погрешностью. Природа этой погрешности связана с ограниченным количеством значащих цифр мантиссы и с необходимостью при выполнении сложения и вычитания размещать разряды с одинаковым весом друг под другом. Рассмотрим появление погрешности на действиях с десятичными числами. Предположим, что величина мантиссы M не превышает 1, а количество разрядов равно 6.

Пусть $a=0.523456*10^2$, $b=0.746879*10^2$. Тогда

$a+b = 1.270335*10^2 \approx 0.127033*10^3$ (сохранены только 6 разрядов). Возникает погрешность, равная $0.0000005*10^2$. При сложении

$a1 = 0.523456*10^4$ и $b=0.746879*10^2$ получим $0,53092479*10^4$, или, оставляя 6 значащих цифр, $0.530924*10^4$. Погрешность при выполнении операции составила $0.00000079*10^4$.

При умножении шестиразрядного числа на шестиразрядное получается двенадцатиразрядное число, в котором следует сохранить только старшие 6 разрядов.

Пусть снова $a=0.523456*10^2$, $b=0.746879*10^2$. Вычислив

$a*b = 0.390958293824*10^4$ и оставив 6 разрядов, получим $0.390958*10^4$, т.е. погрешность $0.000000293824*10^4$.

С ростом количества арифметических операций погрешность накапливается и может достичь такой величины, что результат потеряет смысл. Поэтому при решении вычислительных задач главное – не формула для вычислений, а умение вычислить значение с минимальным количеством операций, контролируя погрешность.

5.4.2. Библиотечные математические функции Java

Класс Math содержит функции с плавающей точкой, которые применяются в алгебре, геометрии и тригонометрии (таблицы 5.2-5.4), а также несколько универсальных методов. В Math определены две константы типа double: E (приблизительно 2,72) и PI (приблизительно 3,14). Класс Math находится в пакете java.lang, который импортируется автоматически во все программы.

Таблица 5.2 – Методы трансцендентных функций

Метод	Описание
`static double sin(double arg)`	Возвращает синус угла, указанного параметром *arg* (в радианах)
`static double cos(double arg)`	Возвращает косинус угла, указанного параметром *arg* (в радианах)
`static double tan(double arg)`	Возвращает тангенс угла, указанного параметром *arg* (в радианах)
`static double asin(double arg)`	Возвращает угол, чей синус указан параметром *arg*
`static double acos(double arg)`	Возвращает угол, чей косинус указан параметром *arg*
`static double atan(double arg)`	Возвращает угол, чей тангенс указан параметром *arg*
`static double atan2(double x, double y)`	Возвращает угол, чей тангенс равен *x/y*.

Таблица 5.3 – Методы экспоненциальных функций

Метод	Описание
`static double exp(double arg)`	Возвращает значение функции e^{arg} (где e = 2.72, основание натурального логарифма)
`static double log(double arg)`	Возвращает значение натурального логарифма `ln(arg)`
`static double pow(double y,double x)`	Возвращается значение степени y^x; например, `pow(2.0,3.0)` возвращает 8.0
`static double sqrt(double arg)`	Возвращает значение квадратного корня из *arg* (т. е. $arg^{1/2}$)

Таблица 5.4 – Методы округляющих функций

Методы	Описание
`static int abs(int arg)`	Возвращает абсолютное значение *arg*
`static long abs(long arg)`	Возвращает абсолютное значение *arg*
`static float abs(float arg)`	Возвращает абсолютное значение *arg*
`static double abs(double arg)`	Возвращает абсолютное значение *arg*
`static double ceil(double arg)`	Возвращает наименьшее целое число, большее или равное *arg*
`static double floor(double arg)`	Возвращает наибольшее целое число, меньше или равное *arg*
`static int max(int x, int y)`	Возвращает максимум из *x* и *y*
`static long max(long x, long y)`	Возвращает максимум из *x* и *y*
`static float max(float x, float y)`	Возвращает максимум из *x* и *y*
`static double max(double x, double y)`	Возвращает максимум из *x* и *y*
`static int min(int x, int y)`	Возвращает минимум из *x* и *y*
`static long min(long x, long y)`	Возвращает максимум из *x* и *y*
`static float min(float x, float y)`	Возвращает максимум из *x* и *y*
`static double min(double x, double y)`	Возвращает максимум из *x* и *y*
`static double rint(double arg)`	Возвращает ближайшее целое значение *arg*
`static int round(float arg)`	Возвращает *arg*, округленный до ближайшего целого `int`-значения
`static long round(double arg)`	Возвращает *arg*, округленный до ближайшего целого `long`-значения

В дополнение к методам, представленным в таблицах 2.2-2.4 в Math определены следующие методы:

```
static double IEEEremainder(double dividend, double divisor)
static double random()
static double toRadians(double angle)
static double toDegrees(double angle)
```

Метод IEEEremainder() возвращает остаток от деления dividend/divisor.

Метод random() предназначен для получения псевдослучайного числа из промежутка от 0 до 1.

Метод toRadians() конвертирует градусы в радианы.

Метод toDegrees() переводит радианы в градусы.

5.5. Порядок выполнения работы

5.5.1. Работа в окне кода BlueJ

Внимание! Каждое из перечисленных ниже заданий демонстрирует особенности работы с действительными числами в Java. Листинг (скриншот окна кода) для каждого задания должен быть приведен в отчете по лабораторной работе. Результаты выполнения каждого задания исполнительной системой Java (после их тщательного осмысления!) и соответствующие выводы также должны быть приведены в отчете.

1) Набрать в окне кода double a1=2.523, b1=25.23, c1=0.2523; . Получить значения переменных a1, b1, c1. Сделать вывод.

2) Набрать в окне кода double a2=-523.428e3; . Получить значение a2.

3) Набрать в окне кода double a3=523.428e-3; . Получить значение a3. Сделать вывод по пунктам 2) и 3).

4) Вывести значения переменных a1, b1, c1 в окно терминала в математическом формате при помощи оператора
System.out.printf ("a1=%e; b1=%e; c1= %e \n", a1, b1, c1); .

5) Вывести значения переменных a2 и a3 в окно терминала в математическом формате и фиксированном формате при помощи оператора
System.out.printf ("a2 = %e = %14.6f; a3 = %e = %14.6f \n", a2, a2, a3, a3);
Сделать вывод по пунктам 4) и 5).

6) Ввести в окне кода:
final double M=0.25;
M=M+5;
Какое сообщение будет выдано и почему? Сделать вывод.

7) Набрать в окне кода:
double a4=0.25;
double a5=0.25;
Получить значения переменных a4 и a5, а затем − значение выражения a4==a5. Какого оно типа?

8) Набрать в окне кода:
double a6=5,1;
double a7=a6+0.1;
Получить значение выражения a7==5.2. Получить значение переменной a7. Прокомментировать результат. Сделать вывод по пунктам 7 и 8.

9) Набрать в окне кода:

double a8=25.25;

float a9=25.25f;

Получить значения переменных a8 и a9, а затем − значение выражения a8==a9.

10) Ввести операторы:

a8=a8+0.1;

a9=(float) (a9+0.1f);

Получить значения переменных a8 и a9, а затем − значение выражения a8==a9. Зачем в последнем операторе потребовалось явное преобразование типа? Сделайте выводы по пунктам 9 и 10.

11) Вычислить каждое из трех приведенных ниже выражений дважды: один раз с типом double переменной x, второй раз с типом float переменной x (чтобы не возникла ошибка двойного определения переменной x, во втором случае можно использовать имя x1 вместо x). Для оценки погрешности вычисления сравнить результаты. В качестве значения x взять свой день рождения, деленный на месяц рождения плюс 0.1. Возведение в степень вычислять при помощи операции умножения.

1. $x^2/2! + x^3/3!+x^4/4!$

Пример:

```
double x=8.0/11.0;
float x1=(float)(8.0/11.0);
x=x*x/2+x*x*x/(2*3)+x*x*x*x/(2*3*4)
x1=x1*x1/2+x1*x1*x1/(2*3)+x1*x1*x1*x1/(2*3*4)
```

2. $(x-1)/(x+2.5) + ((x-3)/(x-3.5))^2$

3. $(x+2.5*x^2-3.7*x^3)/(1.34+2.7x^3+5.21*x^5)$

Сделать вывод.

12) Набрать фрагмент кода:

```
float a=4.0f;
float b=3.5+a;
```

Какое сообщение выдает система и почему? Исправьте ошибку и получите значение b. Сделайте вывод.

13) Набрать фрагмент кода:

```
int g=4, h=8;
double z=0.5+g/h;
double q=0.5+4/8;
double s=0.5+4.0/8.0;
```

Получите значения z, q и s. Объясните результаты. Сделайте вывод.

14) Наберите в окне кода операторы:

```
System.out.println (Integer.toBinaryString(Float.floatToIntBits (0.9375f)));
System.out.println (Integer.toBinaryString(Float.floatToIntBits (-0.9375f)));
System.out.println (Long.toBinaryString(Double.doubleToLongBits (0.9375)));
System.out.println (Long.toBinaryString(Double.doubleToLongBits (-0.9375)));
```

Объясните результат. Сравните с представлением чисел 0.9375 и -0.9375 в формате IEEE-754 для типов float и double, полученным вручную. Представление числа 0.9375 в формате IEEE-754 для типа float приведено на рисунке 2.3. Сделайте выводы.

5.5.2. Разработка программы для выполнения операций с вещественными числами

1) Разработайте алгоритм вычислений и программу согласно варианту задания (таблица 5.5).

2) Обоснуйте выбор типа для вещественных переменных.

3) Проведите отладку программы и испытание на заданных тестовых примерах.

5.6. Варианты заданий

В качестве индивидуального задания на лабораторную работу предлагается разработать программу, реализующую вычисление по формулам, указанным в таблице 2.5 в соответствии с номером варианта. При составлении вариантов заданий использованы методические указания [6].

Вариант задания V необходимо вычислить по формуле V=N%14,
где N – номер студента в списке группы.
Примечание: если N%14 == 0, то V = 14.

Для решения поставленной задачи выполните следующие этапы.

1) Преобразуйте формулы с целью уменьшения количества операций при вычислениях. Упрощение возможно как за счет математических преобразований, так и за счет введения дополнительных переменных для сохранения значений выражений, неоднократно встречающихся в формуле.

2) Для двух заданных вариантов исходных данных вычислите с помощью калькулятора значения, определяемые формулами (переведите калькулятор в режим работы с радианами при вычислении тригонометрических функций). При этом окажется, что одна пара исходных значений не входит в область допустимых значений, т.е. приведет к возникновению ошибки;

3) Составьте алгоритм (схему) программы.

4) В соответствии с алгоритмом составьте программу на языке Java.

5) Запустите программу на выполнение при обоих вариантах значений входных переменных. Проанализируйте полученные результаты (сравните с

результатами расчетов на калькуляторе) и сообщения исполнительной среды Java.

Таблица 5.5 – Варианты заданий

№ п/п	Типы переменных	Варианты исходных данных	Задания
1	*a,b,y,z* – вещественные	а) a = 5,7; b = 0,08 б) a = -0,3; b = 15,1	$y=\frac{e^{-a}+\frac{z+10^3}{\sin z}}{\cos\pi z+\ln b}$, $z=\lvert b-15{,}1\rvert$
2	*a, x,k* – вещественные; *y* – целая	а) x=-3,06; y=3 б) x=0,215; y= –11	$a=\frac{e^{-tg\,\pi k}-\ln\lvert x\rvert}{kx+10^5}$, $k=\frac{y+11}{2}$
3	k, x, i - вещественные; m – целая	а) x=0,02; m=3 б) x= -1,5; m= – 10	$k=\frac{\ln\lvert i+8\cdot10^{-2}\rvert-\ln i}{\cos\,\pi i+e^x}$, $i=\frac{m-1}{m+1}$
4	m, n, p - вещественные; r – целая	а) m =716,2; r = 1 б) m=0,07; r= -11	$n=e^{-\sin m+tg\frac{\pi r}{p}}$, $p=\ln\left\lvert\frac{r^3+10^3}{r^3-1}\right\rvert$
5	d, k, z – вещественные; x – целая	а) k = -0 ,12;x=0 б) k = 1,5; x= -11	$d=\ln\frac{z^2+10^{-3}}{z^2+1{,}6\cdot10^{-2}}$, $z=e^{-k}\,ctg\,kx$
6	m, x, z - вещественные; y – целая	а) x = 1,016; y=11 б) x= -0.2; y=2	$m=\sin arctg\frac{z}{2}-\sin arctg\frac{\pi}{3}$ $z=\frac{\lvert e^{-xy}+10^{-3}\rvert+e^{-xy}}{\pi+\ln xy}$
7	z, x,m – вещественные; c –целая	а) x= -0.02; c= -3 б) x= -1,1; c=2	$z=tg^2\frac{\pi m}{m+10^{-3}}$, $m=e^{-c\lvert x\rvert+\sqrt{cx}}$
8	K - целая r,t,m – вещественные	а) r = 0,07; k = 6 б) r = 0,63; k = -15	$t=\sqrt{\frac{\cos^2\pi r+k\cdot10^{-2}}{\cos^2\pi r+\lvert\frac{m}{k}\rvert}}$, $m=\ln\cos\pi r$
9	x, z –целые m,k – вещественная	а) x = -4; z = 12 б) x = 2 ; z= -6	$m=\frac{\sqrt{\ln k+10^3}-\sqrt{10^3-\ln k}}{x^2+18x-40}$ $k=e^{\pi x}\cos0{,}01z$
10	X – целая m, k,z – вещественные	а) x = 12; m = 0,51 б) x =-7; m= -0,05	$k=tg^2z+ctg^2z$, $z=\frac{e^{\pi x}-e^{-\pi x}}{10^3+\sqrt{\ln mx}}$
11	k – целая; x, b, n – вещественные	а) k =2750 x = 18,1 б) k =-124 x= -0,73	$b=\sqrt{\frac{\sin^2 n+\sin n^2}{\lvert\sin n\rvert+e^{-x}}}$, $n=\ln\frac{\pi}{kx-1{,}6\cdot10^3}$
12	z – целая; m, r, t – вещественные	а) r = 0,12 z =10 б) r = 8,3 z = -116	$m=180\,arctg\frac{e^{5t}-e^{-5t}}{e^{zt}-e^{-zt}}$, $t=\sqrt{\frac{\ln\lvert\pi r\rvert}{10^3+rz}}$
13	a, b, x, y – вещественные	а) x = 1,625; y=0,825 б) x = -2,35; y= 1,115	$a=\cos^2 tg\left(\frac{1}{b}\right)$, $b=\frac{x+\frac{y}{5+\sqrt{x}}}{\lvert y+x\rvert+\sqrt[3]{x}}$
14	a, b, x, y – вещественные	а) x=47,8; y=5,5; в) x =0; y = 2,3	$a=e^{b-1}+tg^2x$, $b=\frac{2\cos(x-\pi/6)}{\frac{1}{x}+\sin^2 y}$

5.7. Пример выполнения индивидуального задания

Постановка задачи: Вычислить значение *a* по формуле (2.1)

$$a = \frac{\sqrt[3]{|y|}}{(x+1)+(x-1)} + \frac{|\sin x|}{\sqrt[x]{|y|}} \qquad (2.1)$$

для $x = 1{,}241$, $y = -0{,}879$.

Проанализируем формулу (1.4) с целью выявления возможности упростить вычисления. Знаменатель первой дроби можно упростить следующим образом:

$$(x+1)+(x-1)=2x \ . \qquad (2.2)$$

Чтобы дважды не вычислять $|y|$, введем дополнительную переменную

$$b = |y| \ . \qquad (2.3)$$

Тогда с учетом соотношений (2.2), (2.3) формулу для вычисления значения переменной *a* можно переписать следующим образом:

$$a = \frac{\sqrt[3]{b}}{2x} + \frac{|\sin x|}{\sqrt[x]{b}} \ .$$

Алгоритм вычислений будет иметь вид, показанный на рисунке 2.5.

Предварительный расчет значения *a* с помощью калькулятора при заданных значениях *x* и *y* дает:

$b = 0{,}879; \quad \sin x = 0{,}9461;$

$$a = \frac{0{,}9579}{2{,}482} + \frac{0{,}9461}{0{,}9013} = 0{,}3859 + 1{,}0497 = 1{,}4357.$$

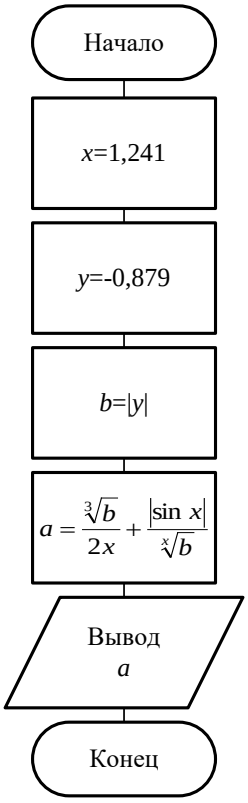

Рисунок 5.5 – Схема алгоритма вычислений

Текст программы на языке Java:

```
public class Lab5_1{
  public static void main (String args []){
     double x,y,b,a;
     x=1.241;
     y=-0.879;
     b=Math.abs(y);
     a=Math.pow(b,1.0/3.0)/(2*x)+Math.abs(Math.sin(x))/Math.pow(b,1.0/x);
     System.out.printf("x=%8.4f;  y=%8.4f;  a=%8.4f \n",x,y,a);
     System.out.println("Значение переменной a в формате IEEE 754: ");
     System.out.println(Long.toBinaryString(Double.doubleToLongBits (a)));
  }
}
```

Результаты выполнения программы приведены на рисунке 5.6.

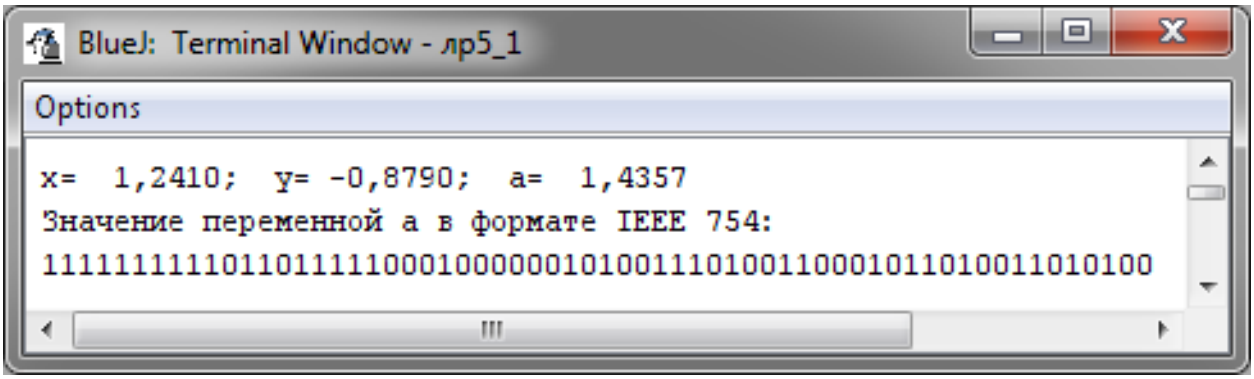

Рисунок 5.6 – Результаты вычислений

Значение переменной **a**, полученное в результате выполнения программы, совпало со значением, рассчитанным при помощи калькулятора.

Внимание! Если в программе вы получили значение, не совпадающее с вашими расчетами (а вы точно уверены, что посчитали на калькуляторе правильно), в процессе отладки программы рекомендуется выводить в окно терминала все промежуточные результаты вычислений для обнаружения логической ошибки в программе. Например, при отладке приведенной выше программы была обнаружена следующая логическая ошибка: выражение для вычисления значения *a* первоначально выглядело так:

```
Math.pow(b,1.0/3.0)/(2*x)+Math.abs(Math.sin(x))/Math.pow(b, x)
```

вместо

```
Math.pow(b,1.0/3.0)/(2*x)+Math.abs(Math.sin(x))/Math.pow(b,1.0/x),
```

что и давало неправильный результат.

5.8. Рекомендации по составлению отчета по лабораторной работе

Отчет должен соответствовать требованиям, изложенным в пункте 2.8, и содержать разделы, перечисленные в пункте 1 данных методических указаний.

5.9. Контрольные вопросы

1) Какие типы Java определяют вещественные числа?
2) Какой формат используется для внутреннего представления вещественных данных? Из каких полей состоит этот формат.
3) Какое вещественное число считается нормализованным в математике, а какое в формате IEEE-754?
4) От чего зависит точность представления вещественных чисел в ЭВМ? От чего зависит диапазон представления вещественных чисел в ЭВМ?
5) Какой из форматов обеспечивает большую точность и диапазон и почему?
6) Как в программе задать литерал типа float?
7) Как задать литерал типа double?
8) Как выглядит действительное число, заданное в естественной форме (с фиксированным числом знаков после запятой) и в научной форме?
9) Как определить вещественную переменную в программе? Чем нужно руководствоваться при выборе ее типа?
10) Как определить вещественную константу в программе?
11) Когда в Java-программе может возникнуть ошибка, связанная с потерей точности? Приведите пример. Для чего используется явное преобразование типа в выражениях?
12) Объясните, почему возникает и накапливается погрешность при выполнении машинных расчетов с действительными данными. Как это влияет, в частности, на результат операции сравнения на равенство (==) действительных переменных.
13) Какой тип с плавающей точкой обеспечивает меньшую погрешность и за счет чего?
14) Какие библиотечные методы вы применили для получения текстового эквивалента представления вещественного числа в форматах IEEE-754 с одинарной и двойной точностью. Какие оговорки нужно сделать по-поводу полученного представления положительного числа? В каких классах-оболочках находятся использованные методы?
15) В каком библиотечном классе определены методы для вычисления математических функций? Нужно ли импортировать в программу этот класс? Какие функции вы использовали в своей программе?

6. ЛАБОРАТОРНАЯ РАБОТА № 5. ОБРАБОТКА ДАННЫХ ТИПА CHAR И BOOLEAN. УСЛОВНЫЕ ВЫРАЖЕНИЯ. ТРОИЧНАЯ УСЛОВНАЯ ОПЕРАЦИЯ JAVA

6.1. Цель работы

Освоить работу с типами boolean и char, научиться применять методы класса Character для анализа символов, научиться составлять сложные условные выражения, изучить синтаксис и научиться применять троичную условную операцию Java.

6.2. Постановка задачи

1) Ознакомиться с принципами хранения и обработки символов и логических значений в Java, а также с принципами построения логических выражений.

2) Выполнить заданные операции над данными типа char в окне кода.

3) Выполнить заданные операции над данными типа boolean в окне кода.

4) Разработать и отладить программу, демонстрирующую использование типов char и boolean, условных выражений, а также троичной условной операции.

6.3. Внеаудиторная подготовка

Для подготовки к лабораторной работе следует изучить краткие теоретические сведения, приведенные в пункте 6.4 методических указаний. Рекомендуется также ознакомиться с [1] (с.48-50, 52-64).

6.4. Краткие теоретические сведения

6.4.1. Тип char

Тип char представляет коды символов Unicode. Код символа – это номер соответствующего символа в таблице Unicode. Код является 16-битовым целым значением (без знака). Дополнительные сведения о символах Unicode можно получить, обратившись к Интернет-ресурсам:

http://www.unicode.org/charts/PDF,

ru.wikipedia.org/wikisagesign.ru./tools/Unicode,

http://www.javaportal.ru/java/articles/ruschars/ruschars.html и др.

Число символов, которые можно закодировать с помощью 16 битов равно 2^{16}=65536 (диапазон значений кода символа ─ от 0 до 65535).

Преимущество Java: набор символов Unicode определяет полный набор интернациональных символов, который может представлять все символы,

находящиеся во всех человеческих языках. Язык java ссоздан для разработки апплетов «всемирного использования»!

Недостаток: Использование Unicode несколько неэффективно для языков, подобных английскому, немецкому, французскому, чьи символы могут легко помещаться в 8 битах. (плата за «глобальную мобильность»).

Стандартный набор символов, известный как ASCII (включает буквы английского алфавита, цифры, символы пунктуации, специальные символы) располагается в интервале значений кодов от 0 до 127 (как обычно), а расширенный восьмиразрядный набор символов ISO-Latin-1 ─ в диапазоне от 0 до 255.

Фрагмент кодовой таблицы, содержащий кириллические символы приведен на рисунке 3.1.

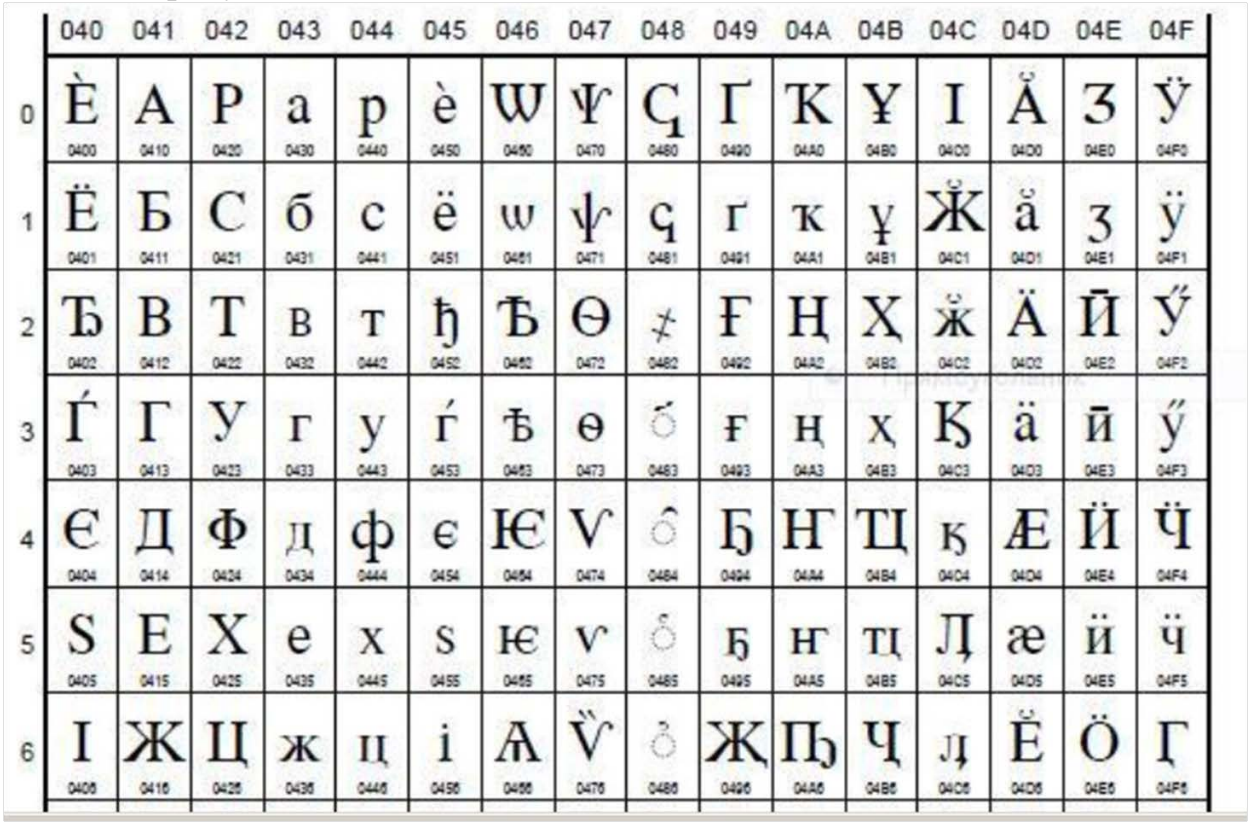

Рисунок 3.1. – Фрагмент таблицы Unicod с кириллическими символами (www.unicode.org)

Тип char в Java по существу является подмножеством типа int, поэтому для типа char определены целочисленные операции, например, для данных типа char допустимо действие код ± целое. В программе программист действует с символами, а компилятор преобразует эти действия в работу с кодами. Следует обратить внимание, что при вычисления выражения, в котором используются целочисленные операции над символами, их коды преобразуются в значения типа int. Соответственно тип значения такого выражения – int. Если это значение присваивается переменной типа char, нужно использовать явное преобразование типа, например,

char ch=(char)('A'+'B'); .

Операции сравнения символов на самом деле сравнивают их коды.

В программе можно использовать переменные и литералы типа char.

Литерал типа char – это символ, заключенный в одиночные кавычки: 'z', '7', '+', '&', 'ю', 'Я' и т.д.

Символ можно задать также его кодом (целым числом в десятичной системе счисления), например char ch=88;

Некоторые «символы» не имеют графического изображения. Такие символы кодируются особо, например, '\n' – новая строка, '\r′ – возврат каретки, '\t′ – табуляция. Некоторые символы имеют специальный смысл – одиночные и двойные кавычки, слэш. Литералы этих символов записываются при помощи символа \ : одиночная кавычка – '\'', двойная кавычка '\"', слэш – '\\'. Можно задать восьмеричный или шестнадцатеричный код символа: '\aaa' (не более трех восьмеричных цифр a) или '\uxxxx' (ровно четыре шестнадцатеричных цифры x). Такой способ задания называется escape-последовательностью. Основные escape-последовательности приведены в таблице 6.1.

Таблица 6.1 – Основные escape-последовательности

Escape-последовательность	Описание
\ddd	Восьмеричный символ (ddd)
\uxxxx	Шестнадцатеричный символ UNICODE (xxxx)
\'	Одиночная кавычка
\"	Двойная кавычка
\\	Обратный слэш
\r	Возврат каретки
\n	Новая строка (известный так же, как перевод строки)
\f	Перевод страницы
\t	Символ табуляции (Tab)
\b	Возврат на один символ (Backspace)

В практике программирования символы принято делить на категории: буквы, цифры, управляющие символы, пробельные символы и другие. Различают также большие и маленькие буквы. Проверка того, что символ принадлежит определенной категории, предусмотрена в классе-оболочке Character. Так, Character.isDigit(x) возвращает true, если x – цифровой символ. Метод isLetter (x) проверяет, буква ли x. Имеются методы isLetterOrDigit(x) (x – буква или цифра?), isLowerCase(x) (x – маленькая буква?), isUpperCase(x) (x – большая буква?), isSpaceChar(x) (x – пробел?).

Метод Character.toUpperCase(x) возвращает большую букву, соответствующую букве − значению x (проверьте в панели кода: Character.toUpperCase('ж')). Метод Character.toLowerCase(x) сопоставляет маленькую букву, соответствующую букве − значению x.

6.4.2. Тип boolean

Булевский тип в Java – это простой тип boolean и класс-оболочка Boolean. Литералы типа boolean – это false и true («ложь» и «истина»). К булевскому типу применимы операции сравнения (== и !=) и операции булевой логики (таблица 6.2).

Таблица 6.2 – Операции булевой логики

Обозначение	Название
&	Логическое И (Logical AND)
|	Логическое ИЛИ (Logical OR)
^	Логическое исключающее ИЛИ (Logical XOR (exclusive OR))
&&	Укороченное И (Shift-ciruit AND)
||	Укороченное ИЛИ (Shift-ciruit OR)
!	Логическое унарное отрицание (Logical unary NOT)
?:	Троичная условная операция (Ternary if-then-else)

Значения булевского типа получаются как результат арифметического сравнения значений других типов и/или как результат операций И (AND − &), ИЛИ (OR − |), Исключающее ИЛИ (XOR − ^), НЕ (NOT − !). Таблица 6.3 демонстрирует правила выполнения этих операций (аналогичны действиям этих операций на битах целых чисел).

Таблица 6.3 – Правила выполнения операций булевой логики

A	B	A&B	A|B	A^B	!A
false	false	false	false	false	true
false	true	false	true	true	true
true	false	false	true	true	false
true	true	true	true	false	false

Короткие логические операции (&& и ||).

Как видно из таблицы 3.4, односимвольная операция ИЛИ приводит к true-результату, когда операнд A – true, независимо от того, каков B. Точно также операция И приводит к false-результату, когда A – false, независимо от того, каков B.

Если использовать двузначные формы (&& и ||) вместо однозначных (& и |), Java вообще не будет выполнять оценку правого операнда, если значение выражения полностью определяется значением левого операнда.

Использование короткой формы стало стандартной практикой в булевской логике, оставляя версии с одиночным символом исключительно для поразрядных операций целыми типами (хотя есть и исключения из этого правила).

Типовое применение булевского типа и операций с ним – хранение результатов проверки сложных условий. Например, (x>2.0)&(x<5.0) будет иметь значение true, если действительное x удовлетворяет неравенству 2.0<x<5.0. Выражение !(x>2.0) истинно для тех же x, что и выражение (x ≤ 2.0).

Результаты булевских операций с константами и булевской переменной x: !true → false, !false → true, x|true → true, x|false → x, x&false → false, x&true → x.

Троичная условная операция.

В Java определена троичная (с тремя операндами) условная операция (?:). Ее синтаксис:

expression1 ? expression2 : expression3

expression1 – выражение, имеющее значение типа boolean.

Если оно true, то вычисляется expression2, иначе (если expression1 − false), вычисляется expression3. Результатом условной операции является значение вычисляемого выражения. Требуется, чтобы expression2 и expression3 возвращали значение одного и того же типа, который не может быть void).

Пример демонстрирует предотвращение деления на нуль благодаря применению условной операции:

ratio = **denom==0 ? 0 : num/denom**; .

Частному ratio присваивается 0, если делитель denom равен нулю, и результат деления в противном случае.

6.5. Порядок выполнения работы

6.5.1. Работа в окне кода BlueJ с типом char

1) Объявите char-переменную и присвойте ей значение литерала – маленькой латинской буквы, например, 'z'. Проверьте значение переменной, нажав Enter.

2) Измените значение переменной, уменьшив или увеличив его на небольшое целое число (здесь нужно использовать явное преобразование типа в char для выражения в правой части оператора присваивания – объясните почему). Проверьте значение переменной и сформулируйте свой вывод.

3) Наберите последовательно в окне кода

'Q'<'F'
'F'<'Q'
'A'>'9'
'X'==88
'Y'=='X'+1

Проанализируйте значения, возвращенные операциями сравнения, и сделайте соответствующие выводы.

4) Проверьте работу методов toUpperCase () и toLowerCase(). Для этого введите в панель кода следующие операторы:

char a1= Character.toUpperCase ('v');
char a2= Character.toUpperCase ('V');
char a3= Character.toLowerCase ('Q');
char a4= Character.toLowerCase ('q');

После ввода каждого оператора проверяйте значение соответствующей переменной. Проанализируйте результаты. Сделайте выводы.

5) Проверьте работу методов isLetter(), isDigit(), isLetterOrDigit() (аргументом методов является символ). Для этого введите в панели кода следующие операторы:

boolean b1=Character.isLetter('A');
b1=Character.isLetter('9');
b1=Character.isDigit('A');
b1=Character.isDigit('9');
b1= Character. isLetterOrDigit ('9');
b1= Character. isLetterOrDigit ('A');
b1= Character. isLetterOrDigit ('&');

После ввода каждого оператора проверяйте значение переменной b1. Проанализируйте результаты. Сделайте выводы.

6) Определите десятичные и шестнадцатеричные коды маленьких и больших букв русского алфавита, набрав

System.out.println((int)буква); или
System.out.printf("%x",(int)буква);
и
System.out.println(Integer.toHexString((int)буква));
например,
System.out.println((int)'Б');
System.out.printf("%x\n", (int)'Б');
System.out.println(Integer.toHexString((int)'Б'));
System.out.println((int)'б');
System.out.printf("%x\n", (int)'б');
System.out.println(Integer.toHexString((int)'б')); .
Сделайте выводы.

6.5.2. Работа в окне кода BlueJ с типом Boolean

1) В панели кода введите:
boolean a=false;
boolean q=true;
a==q
a!=q
a<q
a>q
Сделайте выводы.

2) В панели кода определите числовую переменную x и булевскую переменную b. Вычислите значение b = (x>2.0)&(x<5.0) при x=2.1. Повторите вычисление при x=5.3. Вычислите b = (x>2.0)|(x<5.0) при тех же значениях x. Приведите в отчете результаты и сделанные выводы.

3). Введите булевские переменные c, d, e, f. Вычислите e по формуле e = ! (c & d | true) | (! c | ! d) для всех возможных значений c и d.

4) Вычислите f = (!(c|d)==(!c&!d)) для всех возможных значений c и d.

5) Составьте булевское выражение, которое истинно для значений x, находящихся внутри отмеченных интервалов (рисунок 3.2). Значения границ интервалов выберите самостоятельно. Проверьте в окне кода значение выражения для значений x, находящихся внутри интервалов и за пределами интервалов.

Рисунок 3.1. Интервалы на оси X, в которых выражение должно быть истинно

1) Составьте булевское выражение, истинное для точек с координатами (x,y), лежащих внутри прямоугольника (рисунок 3.3). Расположение прямоугольника на плоскости XOY выберите самостоятельно. Проверьте в окне кода значение выражения для точек, находящихся внутри прямоугольника и за его пределами.

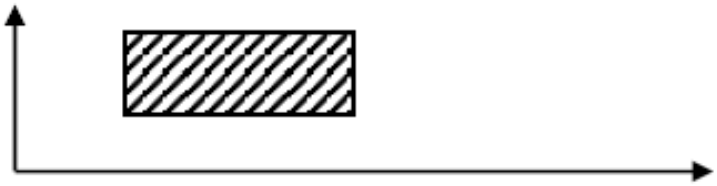

Рисунок 6.2. Область истинности выражения

По пунктам 3)-6) приведите результаты и сделайте выводы.

6.5.3. Разработка программы для выполнения операций с данными типа char и boolean

1) Разработайте программу согласно варианту задания (таблица 6.4).

2) Проведите отладку программы и испытание на достаточном количестве тестовых примеров.

Напомним, что под тестом (тестовым примером) понимается набор входных данных и соответствующий ему набор выходных данных программы. Программист должен разработать для своей программы набор тестов, проверяющий все возможные случаи (ветви) реализации программы.

6.6. Варианты заданий

В качестве индивидуального задания на лабораторную работу предлагается разработать программу, вычисляющую значение символьной переменной *a* в зависимости от значений символьных переменных *c* и *d*. В программе для присвоения значения переменной a предлагается использовать троичную условную операцию. В окне терминала должны быть выведены значения переменных a, c и d в виде символов и в виде кодов символов для каждого тестового примера. Варианты заданий приведены в таблице 7.4.

Вариант задания V необходимо вычислить по формуле

V = N%14 != 0 ? N%14 : 14 ,

где N – номер студента в списке группы.

Таблица 6.4 – Варианты заданий

№ п/п	Правило для вычисления значения переменной *a*
1	a=c+d, если c – цифра, d – буква, или c – знак '&'; a=c+2, в противном случае.
2	a=c–d, если c – буква, d – буква, c>d; a=0470_{16} , в противном случае.
3	a=c, если c='A' или c='F' или c='a' или c='f ', d – цифра; a=d, в противном случае.
4	a=c+10, если c – знак '#' или '@', d – не буква; a='G', в противном случае.
5	a= Character.toUpperCase(d), если c='Y' или '!', d – буква или цифра; a=c, в противном случае.
6	a= Character.toUpperCase(c), если c='y' или 'n', d – не знак '%'; a=d, в противном случае.
7	a=c+20, если c='R' или 'r', d – не цифра и не буква; a=d;
8	a= Character.toLowerCase(c), если c – буква, d – цифра или буква или знак '*'; a=d++, в противном случае.
9	a= Character.toLowerCase(d), если c – цифра или знак '$', d – буква; a=c;
10	a=d—; если c ='@', или c – буква, или d – не цифра; a=c—, в противном случае.
11	a='Y', если c – буква или цифра, d – не буква и не цифра и не знак '#'; a='N', в противном случае.
12	a= Character.toUpperCase(c), если c – буква, но c не является буквами 'A' или 'a', d ='%'; a=d, в противном случае.
13	a=c+d, если c – буква, но не 'R', d – цифра, но не '9'; a='7', в противном случае.
14	a='5', если c='9' или c='0', d – не буква и не '#'; a=c++, в противном случае.

6.7. Рекомендации по составлению отчета по лабораторной работе

Отчет должен соответствовать требованиям, изложенным в пункте 2.8, и содержать разделы, перечисленные в пункте 1 данных методических указаний.

В дополнение к разделам отчета, представленным в пункте 1, опишите все исследования, которые вы провели в окне кода (п. 6.5.1, п. 6.5.2) и сделанные вами выводы.

В задании 3.5.3 особое внимание нужно уделить разделу 5) − тестовые примеры и результаты их обработки вручную. **Разработать тесты для проверки работоспособности всех возможных линий поведения программы**.

6.8. Контрольные вопросы

1) Для чего используется тип char. Сколько битов занимает значение типа char?

2) Что такое Unicode? Как представляется символ в Java? Сколько символов можно представить соответствующим образом?

3) Как можно задать символьный литерал в программе? Назовите 3 способа.

4) Как вывести код символа?

5) Какие операции определены для типа char? Приведите в качестве примера участок кода, требующий явного преобразования типа char в int и int в char.

6) По какому принципу выполняется сравнение символов? Чему равны значения выражений 'A'<'D' , 'C'>'F' , '9'<'A'. Какого они типа?

7) С какими методами, реализующими функции над символами и определенными в классе-оболочке Character, вы познакомились в ходе выполнения данной лабораторной работы? Значения каких типов они возвращают?

8) Для чего используется тип boolean? Какой класс оболочка ему соответствует в Java?

9) Какие значения может принимать тип boolean?

10) Чему равны значения выражений 2>5 , 3<10 ?

11) Какие операции определены для данных логического типа (boolean)?

12) Назовите правила выполнения операций булевой логики.

13) В чем состоит особенность выполнения укороченных операций && и || ?

14) Что из себя представляет условное выражение?

15) Приведите пример сложного условного выражения. Чему равно его значение?

16) Объясните синтаксис троичной условной операции. Приведите пример.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ноутон П. Java 2/ П.Ноутон, Г.Шилдт. – СПб.: БХВ-Петербург, 2007. – 1072 с.
2. Общие требования к текстовым документам. ГОСТ 2.105—95. ЕСКД. М.: «Стандартинформ», 2005. — 28 с.
3. JavaTM 2 Platform Standard Edition 5.0 API Specification. – Электронный ресурс. – Режим доступа: https://docs.oracle.com/javase/1.5.0/docs/api/index.html.
4. ЕСПД. ГОСТ 18.701−90 (ИСО 5807-85). Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – Введ. 1992-01-01. – М: Государственный стандарт Союза ССР.
5. Яшкардин В. IEEE 754 - стандарт двоичной арифметики с плавающей точкой. – Электронный ресурс. – Режим доступа: http://www.softelectro.ru/ieee754.html.
6. Основы языка программирования Pascal: Методические указания к выполнению лабораторных работ, входящих в блок №1 лабораторного практикума по дисциплине «Алгоритмизация и программирование»/ Сост. Д.Н. Старинская, А.А. Кабанов, В.В Захаров. – Севастополь: Изд-во СевГУ, 2016. – 25 с.
7. Простой и понятный самоучитель по Word и Excel. – Электронный ресурс. – Режим доступа: http://www.kavserver.ru/library/wordexcelselftutorial.shtml
8. Microsoft Office 2016. Новейший самоучитель. – Электронный ресурс. – Режим доступа: http://www.kavserver.ru/library/office2016manual.shtml.

ПРИЛОЖЕНИЕ А
Пример оформления титульного листа отчета

Институт информационных технологий и управления
в технических системах

Кафедра информационных технологий и компьютерных систем

ОТЧЕТ
по лабораторной работе № 1
«ВЫПОЛНЕНИЕ ПРОСТЫХ ПРОГРАММ В BLUEJ»
по дисциплине «Программирование. Базовые процедуры обработки информации»

Выполнил студент группы ИВТ/б-11д
Орлов И.В.
Проверил доцент Петров И.И.

Севастополь
2021