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 2
Создание программ с разветвляющейся и циклической структурой.
Управляющие операторы

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 784 Программирование. Базовые процедуры обработки информации. Методические указания к лабораторным работам по дисциплине «Программирование. Базовые процедуры обработки информации» для студентов направлений 09.03.01 – Информатика и вычислительная техника и 09.03.04 – Программная инженерия дневной формы обучения. [В 4 частях]. Часть 2. Создание программ с разветвляющейся и циклической структурой. Управляющие операторы / Министерство образования и науки Российской Федерации, ФГАОУ ВО «Севастопольский государственный университет», Институт информационных технологий и управления в технических системах, кафедра информационных технологий и компьютерных систем ; сост. Т. В. Волкова, Е. С. Владимирова. – Севастополь : [Изд-во СевГУ], 2020.– 32 с. – Текст : непосредственный

Целью методических указаний является оказание помощи в подготовке к лабораторным работам по дисциплине «Программирование. Базовые процедуры обработки информации», предусматривающим разработку и выполнение простых программ с разветвляющейся и циклической 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 № 6. ИССЛЕДОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

2.1. Цель работы

Целью данной работы является исследование разветвляющихся алгоритмов и их программирование с помощью условных операторов языка Java, приобретение начальных навыков тестирования программ.

2.2. Постановка задачи

1) Изучить возможности условных операторов Java.
2) Разработать и отладить программы, демонстрирующие применение условных операторов if и switch.
3) Протестировать программы на контрольных примерах.

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.106-114) и повторить материал, изложенный в [2].

2.4. Краткие теоретические сведения

Часто при выполнении программы порядок действий должен зависеть от результата проверки некоторого условия. При истинности условия должна выполняться одна группа операторов, при ложности – другая. Например, при вычислении корней квадратного уравнения при неотрицательном дискриминанте применяется одна формула, а при отрицательном – другая. На схемах программ проверка условия изображается ромбом (рисунок 2.1, слева). Разделение хода выполнения программы на несколько ветвей называют ветвлением. Ветвление — однократное выполнение одной из двух или более операций, в зависимости от истинности некоторого заданного условия. В структурном программировании ветвление или выбор (ветвление с несколькими ветвями) изображают специальным соединением элементов (рисунок 2.1, справа) [3]. Напомним, движение по линиям сверху вниз и слева направо стрелками не отмечается, а движение справа налево или вверх указывается стрелками.

В языке Java для организации ветвлений предусмотрены:

1) оператор присваивания, использующий троичную условную операцию) :

переменная=условие? выражение1 : выражение2 ;

2) условный оператор (полный и неполный) – if else:

if (условие) оператор1;
else оператор2; *// else может отсутствовать*;

//операторы могут быть простыми и составными

3) многозвенная форма условного оператора − if else if:

```
if (условие1) оператор1;
else if (условие2) оператор2;
else if (условие3) оператор3;
…
else операторN; // else может отсутствовать;
//операторы могут быть простыми и составными
```

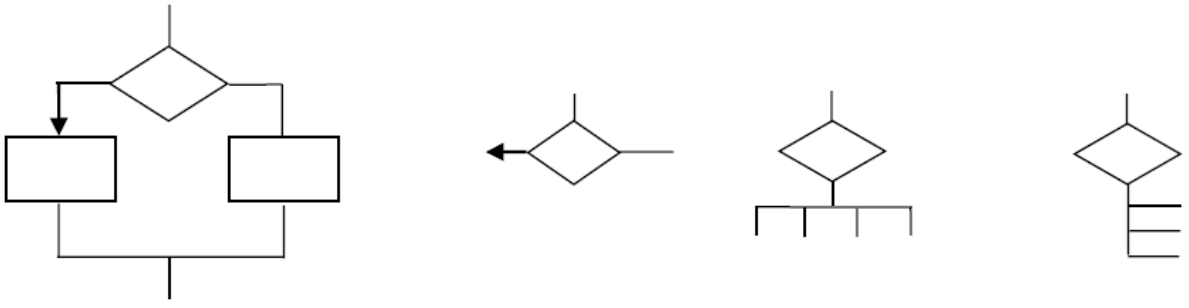
4) оператор-переключатель – switch:

```
switch (выражение) {
    case value1:
      //последовательность операторов 1
      break;

    case value2:
      //последовательность операторов 2
      break;
    …

    case valueN:
      //последовательность операторов N
      break;

    default:
      //последовательность операторов по умолчанию
      // default может отсутствовать
} //Оператор break – необязательный. Если он пропускается,
//выполнение будет продолжено со следующей case-метки.
```



а) б)

Рисунок 2.1 − Структура «ветвление» (а); изображение элементов для организации ветвления с двумя или несколькими альтернативами (б)

2.5. Порядок выполнения работы

1) Изучите пример программы, приведенный в разделе 2.7.1.

2) Составьте алгоритм (схему) программы, демонстрирующей применение оператора if (варианты индивидуальных заданий приведены в таблице 2.1. В программе должно быть предусмотрено задание значений переменных, вывод в окно терминала результатов с пояснениями.

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

4) Разработайте тесты для проверки программы: самостоятельно подберите такие значения исходных данных (переменной x), чтобы, запустив программу, можно было проверить правильность выбора каждой из ветвей алгоритма; рассчитайте на калькудяторе значения выходных переменных (g, z, y) для каждого значения переменной x.

5) Запустите программу на выполнение при трех вариантах значений для переменной x и двух вариантах значений для переменной a. Проанализируйте полученные результаты и сообщения среды Java.

6) Изучите пример программы, приведенный в п. 2.7.2.

7) Составьте алгоритм (схему) программы, демонстрирующей применение оператора switch (варианты индивидуальных заданий приведены в таблице 2.2).

8) Разработайте тесты для проверки программы: самостоятельно подберите такие значения исходных данных, чтобы, запустив программу, можно было проверить правильность выбора каждой из ветвей алгоритма;

9) Проверьте правильность выполнения программы с помощью разработанных тестов. Проанализируйте полученные результаты и сообщения среды Java.

2.6. Варианты заданий

В качестве первого индивидуального задания на лабораторную работу предлагается разработать программу, реализующую вычисление y=g(x)+z(x) при заданном значении x. Формулы для вычисления g и z приведены в таблице 2.1 для каждого варианта.

В качестве второго индивидуального задания на лабораторную работу предлагается разработать программу, реализующую выбор одного из нескольких вариантов действий в зависимости от значения переменной choice. Варианты второго задания приведены в таблице 2.2.

Номер варианта задания V необходимо вычислить по формуле

```
int  V = (N%14 != 0) ? N%14 : 14;
```

где N – номер студента в списке группы.

Таблица 2.1 – Варианты задания 1

Номер варианта	g	z
1	$g=\begin{cases}\dfrac{3x^2}{1+x^2}, & x\le 0\\ \sqrt{1+\dfrac{2x}{1+x^2}}, & x>0\end{cases}$	$z=\begin{cases}3x+\sqrt{1+x^2}, & x<0\\ 2\cos(x)e^{-2x}, & x\in[0,1]\\ 2\sin(3x), & x>1\end{cases}$
2	$g=\begin{cases}\dfrac{3+\sin^2(2x)}{1+\cos^2(x)}, & x\le 0\\ 2\sqrt{1+2x}, & x>0\end{cases}$	$z=\begin{cases}\sqrt{1+\dfrac{x^2}{1+x^2}}, & x<0\\ 2\cos^2(x), & x\in[0;1]\\ \sqrt{1+\lvert 2\sin(3x)\rvert^{1/3}}, & x>1\end{cases}$
3	$g=\begin{cases}\dfrac{3+\sin(x)}{1+x^2}, & x\le 0\\ 2x^2\cos^2(x), & x>0\end{cases}$	$z=\begin{cases}\lvert x\rvert^{\frac{1}{3}}, & x<0\\ -2x+\dfrac{x}{1+x}, & x\in[0,1[\\ \dfrac{\lvert 3-x\rvert}{1+x}, & x\ge 1\end{cases}$
4	$g=\begin{cases}\sqrt{1+2x^2-\sin^2(x)}, & x\le 0\\ \dfrac{2+x}{\sqrt[3]{2+e^{-0.1x}}}, & x>0\end{cases}$	$z=\begin{cases}\dfrac{1+x}{1+x^2}, & x<0\\ \sqrt{1+\dfrac{x}{1+x}}, & x\in[0;1[\\ 2\lvert\sin(3x)\rvert, & x\ge 1\end{cases}$
5	$g=\begin{cases}\sqrt{1+x^2}, & x\le 0\\ \dfrac{1+x}{1+\sqrt[3]{1+e^{-0.2x}}}, & x>0\end{cases}$	$z=\begin{cases}\dfrac{1+x+x^2}{1+x^2}, & x<0\\ \sqrt{1+\dfrac{2x}{1+x^2}}, & x\in[0;1[\\ 2\lvert 0.5+\sin(x)\rvert, & x\ge 1\end{cases}$
6	$g=\begin{cases}\sqrt{1+\lvert x\rvert}, & x\le 0\\ \dfrac{1+3x}{2+\sqrt[3]{1+x}}, & x>0\end{cases}$	$z=\begin{cases}1+\dfrac{3+x}{1+x^2}, & x<0\\ \sqrt{1+(1-x)^2}, & x\in[0;1[\\ \dfrac{1+x}{1+\cos^2(x)}, & x\ge 1\end{cases}$
7	$g=\begin{cases}\dfrac{\sqrt{1+\lvert x\rvert}}{2+\lvert x\rvert}, & x\le 0\\ \dfrac{1+x}{2+\cos^3(x)}, & x>0\end{cases}$	$z=\begin{cases}\dfrac{1+2x}{1+x^2}, & x<0\\ \sin^2(x)\sqrt{1+x}, & x\in[0;1[\\ \sin^2(x)e^{0.2x}, & x\ge 1\end{cases}$

Продолжение таблицы 2.1

Номер варианта	g	z
8	$g=\begin{cases}\sqrt[3]{1+x^2}, & x\le 0\\ \sin^2(x)\dfrac{1+x}{2+\cos^2(x)}, & x>0\end{cases}$	$z=\begin{cases}\dfrac{\lvert x\rvert}{1+x^2}e^{-2x}, & x<0\\ \sqrt{1+x^2}, & x\in[0;1[\\ \dfrac{1+\sin(x)}{1+x}+3x, & x\ge 1\end{cases}$
9	$g=\begin{cases}\dfrac{3+\sin(x)}{1+x^2}, & x\le 0\\ 2x^2\cos^2(x), & x>0\end{cases}$	$z=\begin{cases}\lvert x\rvert^{\frac{1}{3}}, & x<0\\ -2x+\dfrac{x}{1+x}, & x\in[0,1[\\ \dfrac{\lvert 3-x\rvert}{1+x}, & x\ge 1\end{cases}$
10	$g=\begin{cases}\sqrt{1+2x^2-\sin^2(x)}, & x\le 0\\ \dfrac{2+x}{\sqrt[3]{2+e^{-0.1x}}}, & x>0\end{cases}$	$z=\begin{cases}\dfrac{1+x}{1+x^2}, & x<0\\ \sqrt{1+\dfrac{x}{1+x}}, & x\in[0;1[\\ 2\lvert\sin(3x)\rvert, & x\ge 1\end{cases}$
11	$g=\begin{cases}\sqrt{1+x^2}, & x\le 0\\ \dfrac{1+x}{1+\sqrt[3]{1+e^{-0.2x}}}, & x>0\end{cases}$	$z=\begin{cases}\dfrac{1+x}{1+x^2}, & x<0\\ \sqrt{1+\dfrac{x}{1+x}}, & x\in[0;1[\\ 2\lvert\sin(3x)\rvert, & x\ge 1\end{cases}$
12	$g=\begin{cases}\sqrt{1+\lvert x\rvert}, & x\le 0\\ \dfrac{1+3x}{2+\sqrt[3]{1+x}}, & x>0\end{cases}$	$z=\begin{cases}1+\dfrac{3+x}{1+x^2}, & x<0\\ \sqrt{1+(1-x)^2}, & x\in[0;1[\\ \dfrac{1+x}{1+\cos^2(x)}, & x\ge 1\end{cases}$
13	$g=\begin{cases}\dfrac{\sqrt{1+\lvert x\rvert}}{2+\lvert x\rvert}, & x\le 0\\ \dfrac{1+x}{2+\cos^3(x)}, & x>0\end{cases}$	$z=\begin{cases}\dfrac{1+2x}{1+x^2}, & x<0\\ \sin^2(x)\sqrt{1+x}, & x\in[0;1[\\ \sin^2(x)e^{0.2x}, & x\ge 1\end{cases}$
14	$g=\begin{cases}\sqrt[3]{1+x^2}, & x\le 0\\ \sin^2(x)\dfrac{1+x}{2+\cos^2(x)}, & x>0\end{cases}$	$z=\begin{cases}\dfrac{\lvert x\rvert}{1+x^2}e^{-2x}, & x<0\\ \sqrt{1+x^2}, & x\in[0;1[\\ \dfrac{1+\sin(x)}{1+x}+3x, & x\ge 1\end{cases}$

Таблица 2.2 – Варианты задания 2

Номер варианта	Содержание задания
1	Преобразовать рейтинг отеля в число звезд по правилу: 0 – 10 : 1; 11 – 20: 2; 21 – 30: 3; 31 – 40: 4; 41 – 50: 56; иначе: 0 и сообщение «Неправильно задан рейтинг».
2	По первой букве фамилии студента получить номер (его) варианта контрольной работы. Правило преобразования: 'А' – 'Д' : 1; 'Е' – 'К' : 2; 'Л' – 'П' : 3; 'Р' – 'Ф' : 4; 'Х' – 'Щ' : 5; 'Э' – 'Я' : 6; иначе: 0 и сообщение «Неправильно задана фамилия студента».
3	Преобразовать статус футбольной команды в название лиги по правилу: 1 – 5 : "высшая"; 6 – 10 : "первая"; 11 – 15: "вторая"; 16 – 20: "третья"; иначе : если статус в диапазоне от 21 до 50, то "премьер-лига", иначе сообщение «Неправильно задан статус».
4	По первой букве фамилии абитуриента получить номер аудитории, в которой он будет сдавать вступительный экзамен по информатике. Правило преобразования: 'А' – 'Г' : 420; 'Д' – 'Ж' : 421; 'З' – 'Л' : 421а; 'М' – 'О' : 422; 'П' – 'С' : 422а; 'Т' – 'Х' : 423; 'Ц' – 'Щ' : А603; 'Э' – 'Я' : В509; иначе: 0 и сообщение «Неправильно задана фамилия абитуриента».
5	Сообщить студенческим группам время получения учебников в библиотеке университета. Правило преобразования: 11 – 13 : ИВТ/б-11-о; 14 – 15: ИВТ/б-12-о; 16 – 17: ИВТ/б-13-о; иначе сообщение «Неправильно задано время».
6	По числу сотрудников получить характеристику отдела, в котором они работают. Правило преобразования: 1 – 3 : "небольшой"; 4 – 8 : "средний"; 9 – 12: "большой"; 13 – 15: "очень большой"; иначе : если число сотрудников больше нуля, то сообщение «Превышено максимально возможное число сотрудников», иначе сообщение «Неправильно задано число сотрудников».
7	По категории водителя определить наиболее сложный вид транспортного средства, которым он может управлять. Правило преобразования: 'М' : мопеды и квадроциклы; 'А' : мотоциклы; 'В' : легковые автомобили; 'С' : грузовики; 'D' : автобусы; иначе: сообщение «Неправильно задана категория водителя».
8	Зашифровать букву русского алфавита ее порядковым номером. Правило преобразования: 'А' : 1 ; 'В' : 2 ; … 'Я' : 33; иначе сообщение «Неиспользуемый символ».
9	Зашифровать цифру буквой латинского алфавита. Правило преобразования: 1: 'A'; 2: 'B'; 3: 'C'; …; 9: 'H'; 0:'Z'; иначе: сообщение «Неправильно задана цифра».

Продолжение таблицы 2.2

Номер варианта	Содержание задания
10	Преобразовать температуру воздуха (в градусах по Цельсию) в характеристику погоды холодного времени года по правилу: -34 – -43 : "жестоко морозная"; -23 – -33: "сильно морозная"; -13 – -22: "значительно морозная"; -4 – -12: "умеренно морозная"; -1 – -3: "слабо морозная"; 0 – 2: "оттепель"; иначе : если температура меньше нуля, то «крайне морозная», иначе сообщение «Вот и весна пришла ;-)».
11	Преобразовать температуру воздуха (в градусах по Цельсию) в характеристику погоды теплого времени года по правилу: 3 – 7 : "холодная"; 8 – 12: "прохладная"; 13 – 20: "теплая"; 21 – 28: "очень теплая"; 29 – 35: "жаркая"; 36 – 42: "очень жаркая"; иначе : если температура меньше трех градусов, то сообщение «Вот и зима пришла ;-)», иначе – «крайне жаркая».
12	Охарактеризовать тип преподаваемой дисциплины в зависимости от числа кредитов. Правило преобразования: 1 – 2 : "ознакомительная"; 3 – 4: "общеобразовательная"; 5 – 7: "специализированная"; 8 – 9: "углубленно изучаемая"; иначе : сообщение «Неправильно задано число кредитов».
13	По номеру дня (числу месяца) определить декаду месяца, в которую попадает этот день. Правило преобразования: 1 – 10: "первая"; 11 – 20: "вторая"; 21 – 31: "третья"; иначе : сообщение «Неправильно задано число».
14	Выбрать действие, соответствующее значению выражения (a % 5) (переменная a – целая). Правило выбора: 0 – 1: a ++; 2 – 3 : a --; 4: a+=4. Вывести исходное и результирующее значение a. Другие переменные в программе использовать нельзя.

Примечание. В вариантах 2 и 4 в качестве исходного данного используйте одиночный символ (получать первую букву из фамилии мы научимся позже).

2.7. Примеры программ

2.7.1. Пример выполнения первого индивидуального задания

Рассмотрим следующую постановку задачи: составить программу вычисления значения y=g(x)+z(x) при заданном значении x, где g(x) и z(x) − кусочно-заданные функции:

$$g = \begin{cases} \sqrt[3]{1+x^2}, & x \le 0 \\ \sin^2(x)\dfrac{1+x}{2+\cos^2(x)}, & x > 0 \end{cases} \qquad z = \begin{cases} 2e^{-2x} & при\ x < -0.2; \\ \dfrac{1+x^3}{(1+x)} & при\ -0{,}2 \le x \le 0{,}8; \\ 1+\ln(1+x) & при\ x > 0{,}8. \end{cases}$$

Алгоритм вычислений будет иметь вид, показанный на рисунке 2.2. На этом рисунке изображены также пути реализации алгоритма при трех вариантах исходных данных: x=-5.3; x=-0.1; x=1.0. Видим, что эти тестовые примеры покрывают все ветви алгоритма.

Данные, полученные в результате расчетов на калькуляторе:
при x=-5.3, g=3.0754, z=80269.6748, y=80275.7503;
при x=-0.1, g=1.0033, z=1.1099, y=2.1133;
при x=1.0, g=0.6178, z=1.6931, y=2.3110.

Текст программы на Java:

```
public class lab6_1_1 {
    public static void main (String[ ] args){
        double x = 1, g, z, y;
        if (x <= 0) g = Math.cbrt(1 + x * x);
        else g = Math.pow(Math.sin(x),2) * (1 + x)/
                    (2+Math.pow(Math.cos(x),2));
        if (x < -0.2) z = 2 * Math.exp(-2 * x);
        else if (x > 0.8) z = 1 + Math.log(1 + x);
        else z = (1 + Math.pow(x,3)) / (1 + x);
        y = g + z;
        System.out.printf("x = %.5f, g = %.5f, z = %.5f, y = %.5f\n",
                                x, g, z, y);

    } //main
} // class
```

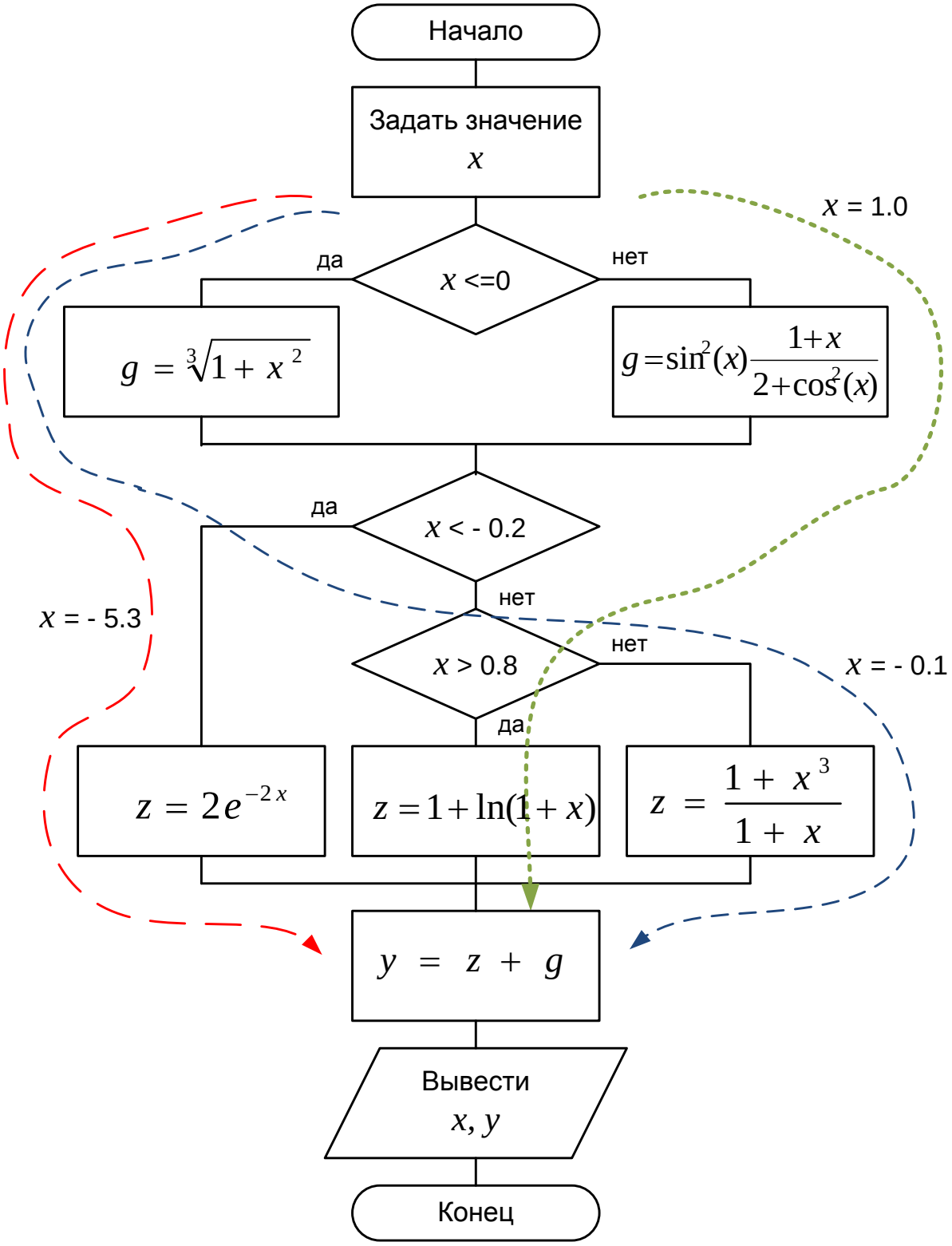


Рисунок 2.2 – Схема алгоритма и тестирования

Второй вариант той же программы использует троичную условную операцию для вычисления значения переменной g, а также импортирование статических функций класса Math (это позволяет обращаться к соответствующей функции просто по ее имени, не указывая перед ним имя класса «Math»):

```
import static java.lang.Math.*;
public class lab6_1_2 {
    public static void main (String[] args){
        double x = 1, g, z, y;
        g = (x <= 0) ? cbrt(1 + x * x) :
            pow(Math.sin(x), 2) * (1 + x)/
                (2 + pow(Math.cos(x), 2));
        if (x < -0.2) z = 2 * exp(-2 * x);
        else if (x > 0.8) z = 1 + log(1 + x);
        else z = (1 + pow(x,3)) / (1 + x);
        y = g + z;
        System.out.printf("x = %.5f, g = %.5f, z = %.5f, y = %.5f\n",
                                x, g, z, y);
    } //main
} // class
```

Результаты выполнения программы для различных значений x представлены на рисунке 2.3.

Значения переменных, полученные в результате выполнения программы, совпали со значениями, рассчитанными 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.

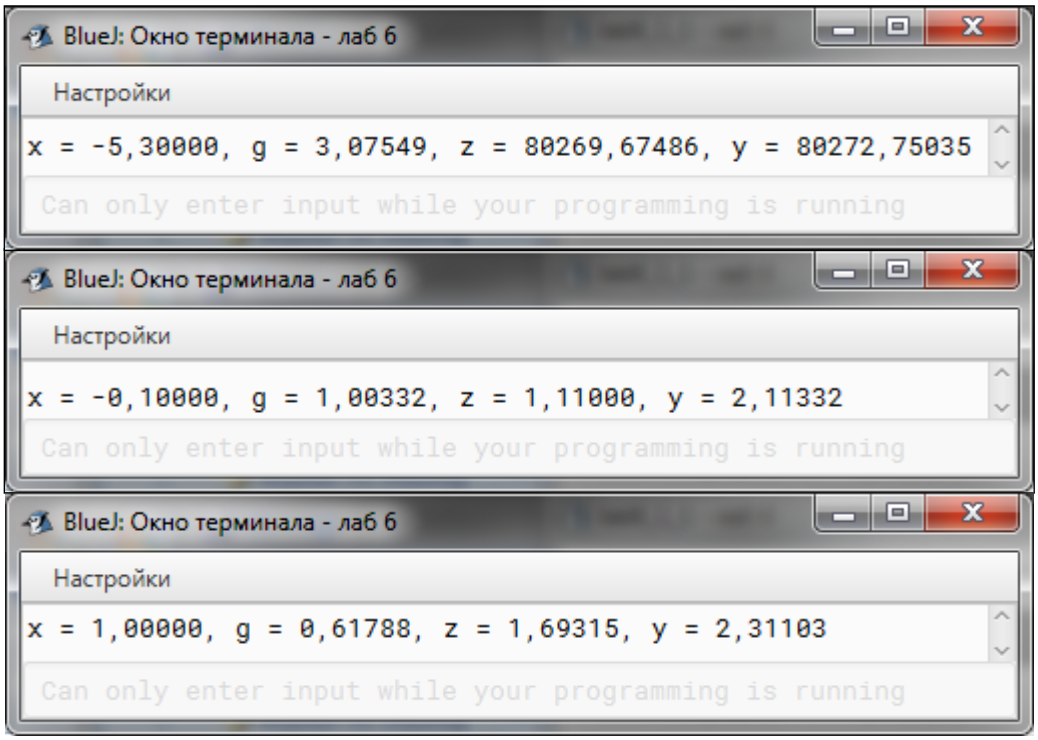


Рисунок 2.3 – Результаты выполнения программы

2.7.2. Пример выполнения второго индивидуального задания

Разработать программу перевода оценки по 100-балльной шкале в оценку по системе ECTS (*англ.* European Credit Transfer and Accumulation System — Европейская система перевода и накопления баллов) по правилу:
90 – 100 баллов : A; 82 – 89 баллов: B; 74 – 81 балл: C; 68 – 73 балла: D; 60 – 68 баллов: E; 0 – 59 баллов: F.

Выбор варианта в программе осуществить с помощью оператора switch.

Схема алгоритма перевода оценки изображена на рисунке 2.4.

Текст программы.

```
public class Lab6_2 {
   public static void main(String args[]){
      int ball = 90; //число баллов
      char ects; //оценка по системе ECTS
      switch (ball){
        case 90: case 91: case 92: case 93: case 94: case 95:
        case 96: case 97: case 98: case 99: case 100:
           ects = 'A';
           break;
        case 82: case 83: case 84: case 85: case 86: case 87:
        case 88: case 89:
           ects = 'B';
           break;
```

```
            case 74: case 75: case 76: case 77: case 78: case 79:
            case 80: case 81:
               ects = 'C';
               break;
            case 68: case 69: case 70:  case 71: case 72: case 73:
               ects = 'D';
               break;
            case 60: case 61: case 62:  case 63: case 64: case 65:
            case 66: case 67:
               ects = 'E';
               break;
            default: ects =(ball<0 || ball > 100) ? '#' : 'F';
         }
         System.out.printf("Число баллов: %d; оценка по ECTS: %c\n", ball, ects);
      }
}
```

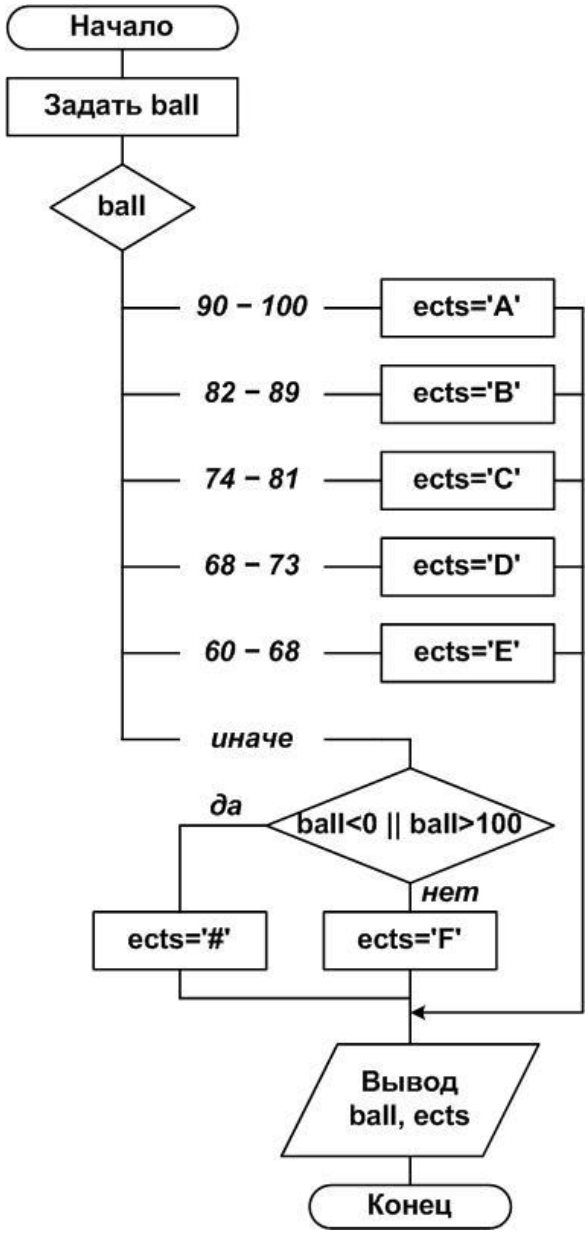


Рисунок 2.4 – Схема алгоритма перевода балльной оценки в оценку ECTS

Результаты работы программы при различных значениях переменной ball приведены на рисунке 2.5

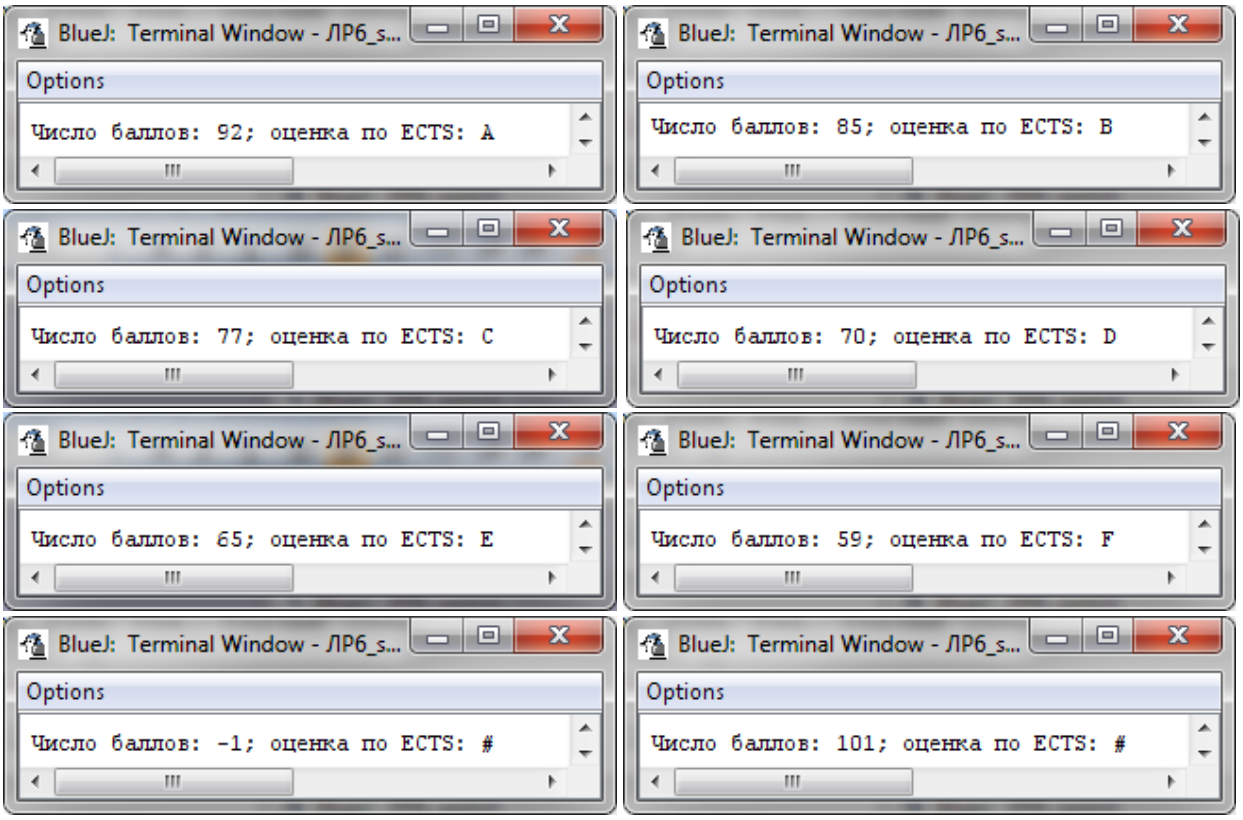


Рисунок 2.5 − Результаты работы программы при различных значениях переменной ball

2.8. Контрольные вопросы

1) Изобразите синтаксическую диаграмму полного и неполного условного оператора. Как эти конструкции изображаются на схемах алгоритмов? Объясните их работу.
2) Поясните термин «логическое выражение». Значение какого типа оно возвращает, какие логические операции в нем могут быть использованы? Приведите примеры сложных логических (условных) выражений. Для чего в условных операторах используются логические выражения (условия)?
3) Что представляет собой составной оператор (блок операторов)? Для чего он используется?
4) В каком случае условный оператор можно заменить оператором присваивания, использующим троичную условную операцию?

5) Перечислите основные блоки, применяемые в схемах разветвляющихся алгоритмов.
6) Опишите синтаксис и алгоритм работы многозвенной формы оператора if: if else if.
7) Опишите синтаксис и алгоритм работы оператора switch? В каких случаях его целесообразно применять? Для чего в ветвях case оператора switch применяется оператор break? Что будет, если в какой-либо ветви его опустить?
8) Для чего в операторе switch применяется элемент default? Можно ли его опустить?
9) Как вы осуществляли тестирование программы? Что такое тест? Сколько тестов вы использовали и почему?
10) Сколько будет **2 * 2 == 4** ? ;-)

3. ЛАБОРАТОРНАЯ РАБОТА № 7. ИССЛЕДОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

3.1. Цель работы

Целью данной работы является изучение назначения и приемов использования операторов циклов Java, исследование циклических алгоритмов и программ, применяемых для решения типовых задач вычисления значения функции по рекуррентной формуле при заданном значении аргумента и табулирования функции на заданном интервале.

3.2. Постановка задачи

1) Ознакомиться с теоретическими сведениями о рекуррентных формулах арифметических корней, приведенными в пункте 3.4 методических указаний, а также с возможностями построения циклических программ в Java.

2) Разработать методы для вычисления функции арифметического корня по рекуррентной формуле согласно варианту задания и для табулирования этой функции на заданном интервале.

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.115-125).

3.4. Краткие теоретические сведения

3.4.1. Операторы циклов

Подавляющее большинство программ содержат циклы. Цикл – это последовательность операторов, которая выполняется несколько раз подряд.

Язык Java поддерживает три оператора цикла.

1) Оператор цикла с предусловием (while) имеет следующий синтаксис:

while (условие) оператор;

Если оператор составной:

```
while (условие) {
    //тело цикла – последовательность операторов
}
```

Выполнение операторов в теле цикла повторяется, пока <условие> (логическое выражение) истинно.

Тело цикла может не выполниться ни разу, если <условие> изначально ложно.

2) Оператор цикла с постусловием (do while) имеет следующий синтаксис:

do оператор; while (условие);

Если оператор составной:

do{

//тело цикла – последовательность операторов

} while (условие);

Выполнение операторов в теле цикла повторяется, пока <условие> (логическое выражение) истинно.

Тело цикла выполняется хотя бы один раз, т.к. значение <условия> (логического выражения) проверяется в конце цикла.

3) Оператор цикла с управляющей переменной (параметром цикла) имеет следующий синтаксис:

for (инициализация; условие; итерация) оператор;

Если оператор, выполняемый в теле цикла, составной:

for (инициализация; условие; итерация){

//тело цикла – последовательность операторов

}

В начале работы цикла выполняется выражение <инициализация>. В общем случае это выражение устанавливает значение *переменной управления циклом*, которая действует как счетчик. Выражение инициализации выполняется только один раз.

Затем оценивается условное выражение <условие>, которое обычно сравнивает переменную управления циклом с каким-либо граничным значением. Если значение условного выражения − true, то выполняются операторы тела цикла, если false, цикл заканчивается.

Далее выполняется часть цикла <итерация>. Обычно это выражение, которое осуществляет инкрементные или декрементные операции с переменной управления циклом.

Каждый проход цикла с новым значением управляющей переменной называется итерацией.

В каждом проходе цикла сначала оценивается условное выражение <условие>, потом выполняется тело цикла и затем – выражение <итерация>. Этот процесс выполняется до тех пор, пока значение выражения <условие> не станет false [3].

В отличие от других языков программирования в Java переменная, управляющая циклом может иметь вещественный тип.

Циклические процессы, реализуемые вышеназванными операторами цикла, изображены на рисунке 3.1.

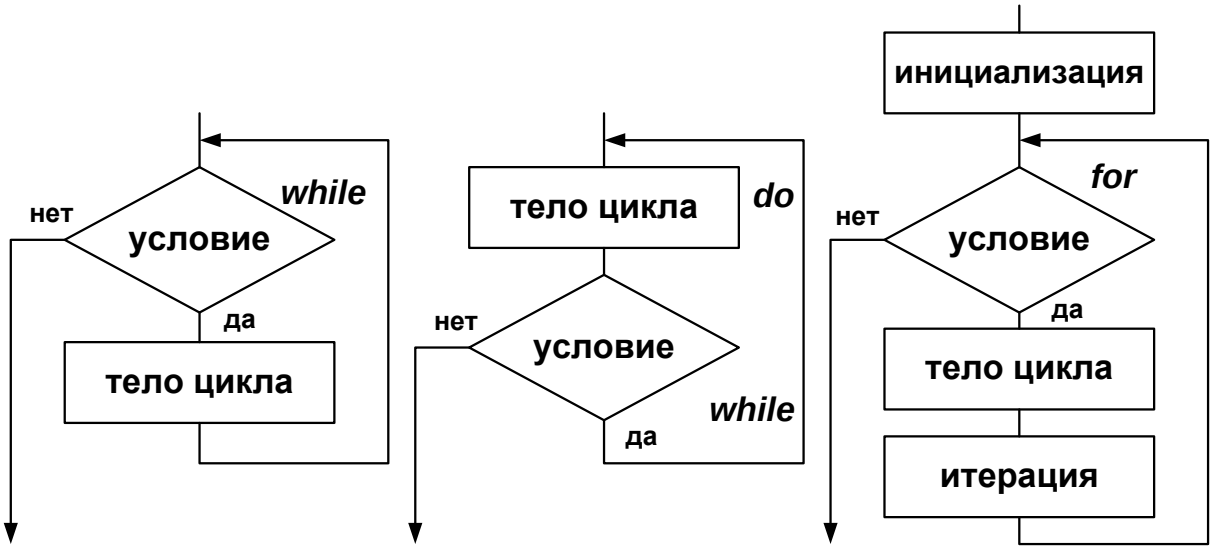


Рисунок 3.1 - Циклы Java

3.4.2. Рекуррентные формулы вычисления значений функций

В различных алгоритмах численного решения задач часто используются итерационные методы. **Метод итераций (метод последовательных приближений)** – метод решения математических задач с помощью построения последовательности, сходящейся к искомому решению, начиная с некоторого начального приближения. При этом члены последовательности вычисляются повторным применением какой-либо операции (итерациями).

Иными словами, итерационные методы – это методы, в которых точное решение может быть получено лишь в результате бесконечного повторения единообразных (как правило, простых) действий.

В итерационных алгоритмах часто применяются так называемые рекуррентные формулы. Рекуррентные формулы служат для получения последовательности значений. При этом каждое следующее значение вычисляется по предшествующим значениям с помощью одной и той же формулы.

Рекурентная формула — формула вида $a_n = f(n, a_{n-1}, a_{n-2}, ..., a_{n-p})$, $n \geq p+1$, выражающая каждый член последовательности a_n ( ) через p предыдущих членов.

В качестве конкретного примера применения такого подхода приведем *правило Ньютона вычисления арифметических корней*.

Пусть x – заданное положительное число, а $n \geq 2$ – заданный натуральный показатель корня. Ставится задача вычислить вещественное значение

$$y = \sqrt[n]{x} \qquad (3.1)$$

путем последовательных приближений с помощью арифметических операций.

Согласно правилу Ньютона процесс приближений к $\sqrt[n]{x}$ определяется формулой

$$y_{k+1} = y_k - \frac{y_k^n - x}{n y_k^{n-1}} \qquad (3.2)$$

или в другом виде

$$y_{k+1} = \frac{1}{n}\left((n-1)y_k - \frac{x}{y_k^{n-1}}\right), \qquad (3.3)$$

где $k = 0, 1, 2, \ldots,$ а $y_0 > 0$ задается.

Еще из глубокой древности известен частный случай правила Ньютона – *процесс Герона*

$$y_{k+1} = \frac{1}{2}\left(y_k - \frac{x}{y_k}\right), \qquad (3.4)$$

применяемый для извлечения квадратных корней $y = \sqrt{x}$.

3.5. Порядок выполнения работы

1) Разработайте программу согласно варианту задания (таблицы 3.1, 3.2).

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.

3.6. Варианты заданий

Вариант задания V необходимо вычислить по формуле

```
int  V = (N%14 != 0) ? N%14 : 14;
```

где N – номер студента в списке группы.

Данные варианта задания нужно взять из таблиц 3.1 и 3.2.

В качестве индивидуального задания на лабораторную работу предлагается разработать класс, включающий несколько методов.

1) Первый метод должен осуществлять вычисление значения функции (арифметического корня) средствами класса Math.

2) Второй метод должен осуществлять вычисление арифметического корня по рекуррентной формуле для заданного значения *x*.

Метод реализует циклический алгоритм, в теле которого пересчитывается по рекуррентной формуле значение y_k при k=1,2,3,…,n. Согласно теории рекуррентных вычислений с ростом k значение y_k приближается к точному значению y. В программе пересчет y_k должен начаться с $y_0 = x$ и повториться n раз. Значения x и n должны передаваться в метод в качестве параметров. На

каждом шаге цикла должен осуществляться вывод y_k и ошибки вычисления по рекуррентной формуле $\varepsilon = |y_{точное} - y_k|$.

Запустите метод на выполнение при двух значениях x из диапазона $[x_{\min}, x_{\max}]$, заданного в таблице 3.1. Пример контекстного меню BlueJ для

Таблица 3.1 – Варианты заданий

№ варианта	Функция	Рекуррентная формула	Исходные данные			
			n	$x_{\min}$	$x_{\max}$	Δx
1	$y = \frac{1}{\sqrt{x}}$	$y_k = 0{,}5 \cdot y_{k-1}(3 - x y_{k-1}^2)$	12	0,4	1,4	0,1
2	$y = \sqrt[4]{x^3}$	$y_k = -0.25 y_{k-1} + (y_{k-1} + x^3 / 4 y_{k-1}^3)$	12	0,2	2,6	0,8
3	$y = \sqrt[3]{x}$	$y_k = 1/3 \cdot (2 y_{k-1} + x / y_{k-1}^2)$	12	0,7	1,6	0,3
4	$y = \sqrt[3]{x^2}$	$y_k = y_{k-1} + 1/3 \cdot (x^2 / y_{k-1}^2 - y_{k-1})$	11	0,9	2,4	0,5
5	$y = \sqrt{x}$	$y_k = y_{k-1}(3/2 - y_{k-1}^2 / 2x)$	12	0,8	2,4	0,4
6	$y = \sqrt{x}$	$y_k = \frac{y_{k-1}(y_{k-1}^2 + 3x)}{3 y_{k-1}^2 + x}$	10	0,5	2,1	0,4
7	$y = \sqrt[3]{x}$	$y_k = \frac{1}{2} y_{k-1} + \frac{1{,}5x}{2 y_{k-1}^2 + x / y_{k-1}}$	10	0,1	2,5	0,6
8	$y = \sqrt[3]{x}$	$y_k = y_{k-1} + 1/3 \cdot (x / y_{k-1}^2 - y_{k-1})$	11	0,7	2,2	0,5
9	$y = \frac{1}{\sqrt[3]{x}}$	$y_k = y_{k-1} + 1/3 \cdot (y_{k-1} + x y_{k-1}^4)$	10	0,8	2,0	0,2
10	$y = \frac{1}{\sqrt[3]{x}}$	$y_k = 1/3 \cdot y_{k-1}(4 - x y_{k-1}^3)$	12	0,9	3,0	0,7
11	$y = \sqrt[4]{x}$	$y_k = y_{k-1} - 0{,}25(y_{k-1} - x / y_{k-1}^3)$	10	1,1	2,3	0,3
12	$y = \frac{1}{\sqrt[4]{x}}$	$y_k = y_{k-1} + 0.25(y_{k-1} - x y_{k-1}^5)$	12	0,9	3,0	0,7
13	$y = \sqrt[5]{x}$	$y_k = y_{k-1} - \frac{y_{k-1}^5 - x}{5 y_{k-1}^4}$	10	0,5	1,2	0,1
14	$y = \sqrt[5]{x}$	$y_k = 0.2\,(4\, y_{k-1} + x / y_{k-1}^4)$	12	0,6	2,0	0,2

запуска методов, реализованных в классе, приведен на рисунке 3.2. Для каждого значения x проанализируйте полученные значения последовательных приближений y_k, вычисленных по рекуррентной формуле, к точному значению y. Сколько итераций потребовалось для получения значения y_k, равного y с точностью до 6 знака? От чего зависит значение необходимого числа итераций?

3)Третий метод должен осуществлять табулирование функции арифметического корня на заданном интервале.

Таблица 3.2 – Виды циклов, подлежащих реализации

Номер варианта	Вычисление корня по рекуррентной формуле	Табулирование функции	Номер варианта	Вычисление корня по рекуррентной формуле	Табулирование функции
1	while	for	8	do while	for
2	do while	for	9	for	while
3	for	while	10	while	for
4	while	for	11	do while	for
5	do while	for	12	for	while
6	for	while	13	do while	for
7	while	for	14	for	while

Под табулированием функции понимают вычисление значения функции при всех значениях аргумента x, изменяющихся с шагом Δx в пределах от $x_{\min}$ до $x_{\max}$.

Для вычисления значения функции третий метод будет вызывать первый метод и второй метод. Вывод промежуточных значений y_k во втором методе предлагается исключить (***закомментировать***). Третий метод должен осуществлять вывод точного значения функции $y_{точное}$ (считаем точным значение функции, вычисленное средствами модуля Math) и приближенного

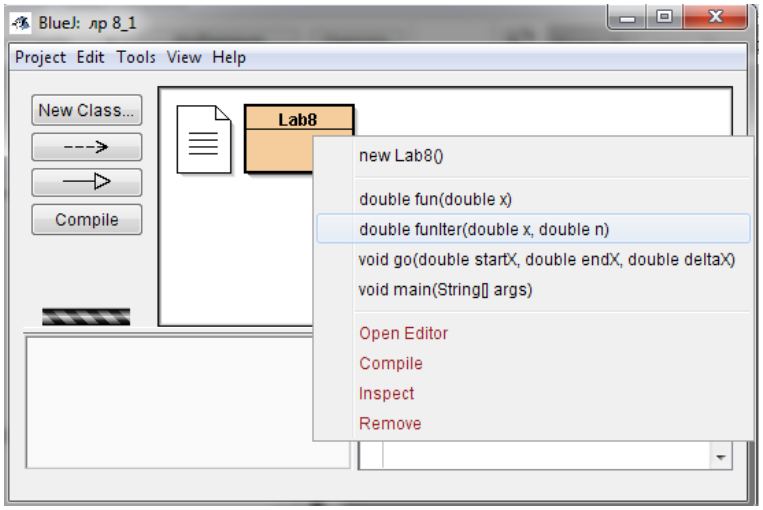


Рисунок 3.2 – Контекстное меню запуска методов проекта в BlueJ

значения y_n (вычисленного по рекуррентной формуле), а также ошибки $\varepsilon = |y_{точное} - y_n|$ для каждого значения x.

Запустите метод на выполнение при значениях n, $x_{\min}$, $x_{\max}$, Δx, заданных в таблице 3.1 и при другом наборе значений.

4) Четвертый метод − main() − должен осуществлять табулирование функции последовательно на нескольких интервалах с различным значением Δx.

Во всех вариантах начальное приближение $y_0 = x$.

3.7. Пример программы

Необходимо составить программу для приближенного вычисления значения функции $y = \sqrt{x}$ с помощью рекуррентной формулы $y_{k+1} = \frac{1}{2}\left(y_k - \frac{x}{y_k} \right)$, для за данного *x*. Алгоритм метода вычисления корня по рекуррентной формуле можно представить с помощью схемы, показанной на рисунке 3.3. Схема алгоритма табулирования функции на заданном интервале показана на рисунке 3.4.

Текст программы будет иметь следующий вид.

```
public class Lab7{
  public static double fun (double x){
     double y = Math.sqrt(x);
     return y;
  }

  public static double funIter (double x, int n){
     double y = x;
     for (int i = 1; i <= n; i++){
        y = (y + x / y) / 2;
        System.out.printf ("x=%10.6f, y=%10.6f, i=%4d\n", x, y, i);
      }
      return y;
  }

public static void go (double startX, double endX, double deltaX){
       double x = startX,yt,yn,e;
       int n = 10;
       while (x <= endX){
          yt = fun(x);
          yn = funIter(x,n);
          e = Math.abs(yt-yn);
          System.out.printf ("x=%10.6f, yt=%10.6f, yn=%10.6f, e=%10.6f\n",
                                x,yt,yn,e);
          x = x + deltaX;
       }
  }

  public static void main (String[ ] args){
       System.out.println ("Начало табулирования");
       go(10,20,2);
       System.out.println ("Табулирование окончено");
       System.out.println ("Начало табулирования");
       go(2.5,3.5,0.1);
       System.out.println ("Табулирование окончено");
    } //main
} // class
```

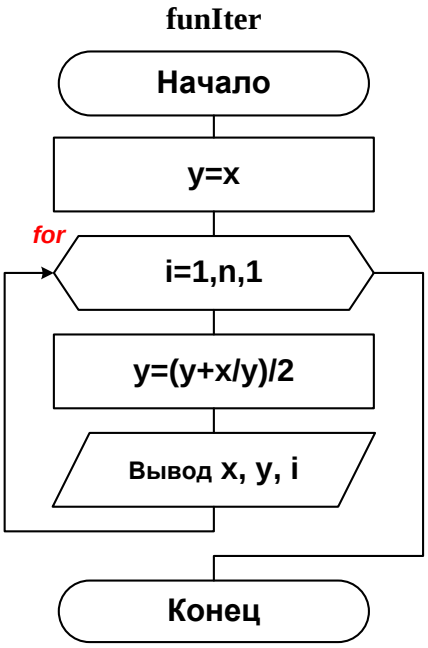


Рисунок 3.2 – Алгоритм вычисления корня по рекуррентной формуле

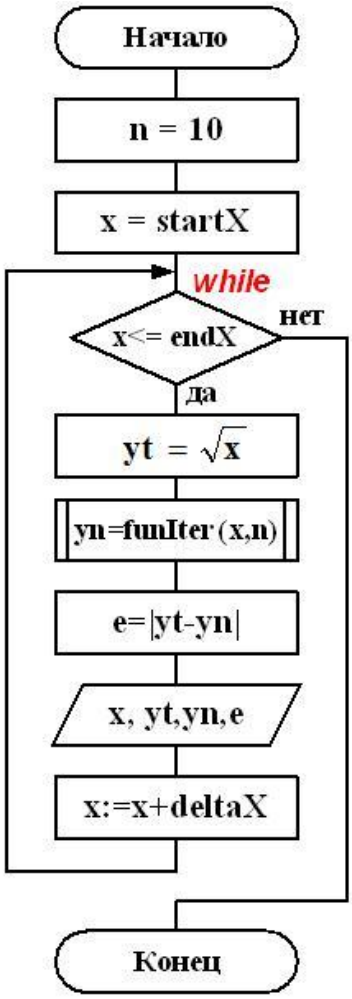


Рисунок 3.3 – Алгоритм табулирования функции

Результаты вызова метода funInt (10,10) приведены на рисунке 3.5.
Результаты вызова метода funInt (500000,15) приведены на рисунке 3.6.

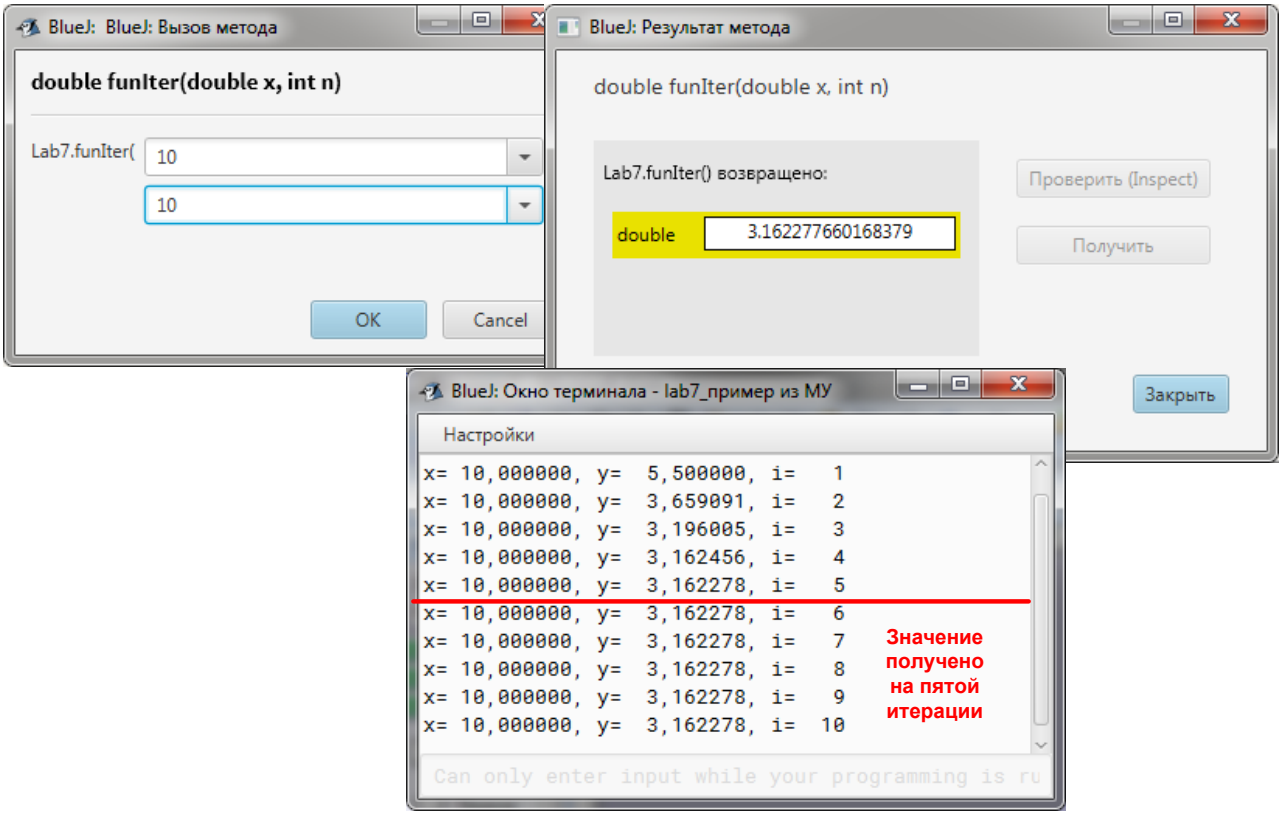


Рисунок 3.5 – Результаты вызова метода funIter (10,10)

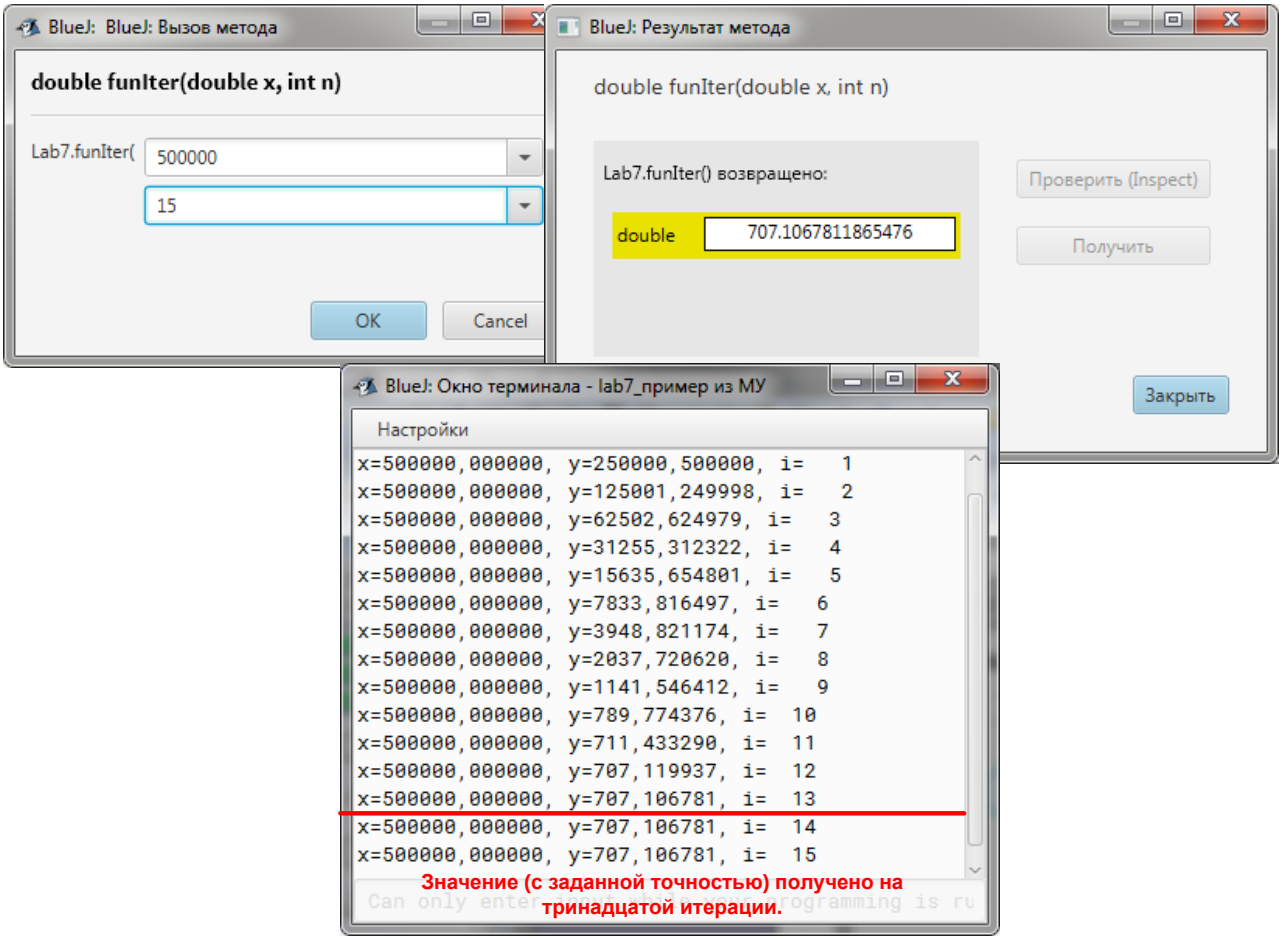


Рисунок 3.6 – Результаты вызова метода funInt (500000,15)

Результаты работы программы приведены на рисунке 3.7 (без вывода приближений, полученных в методе funIter, – соответствующий оператор вывода «закомментирован»).

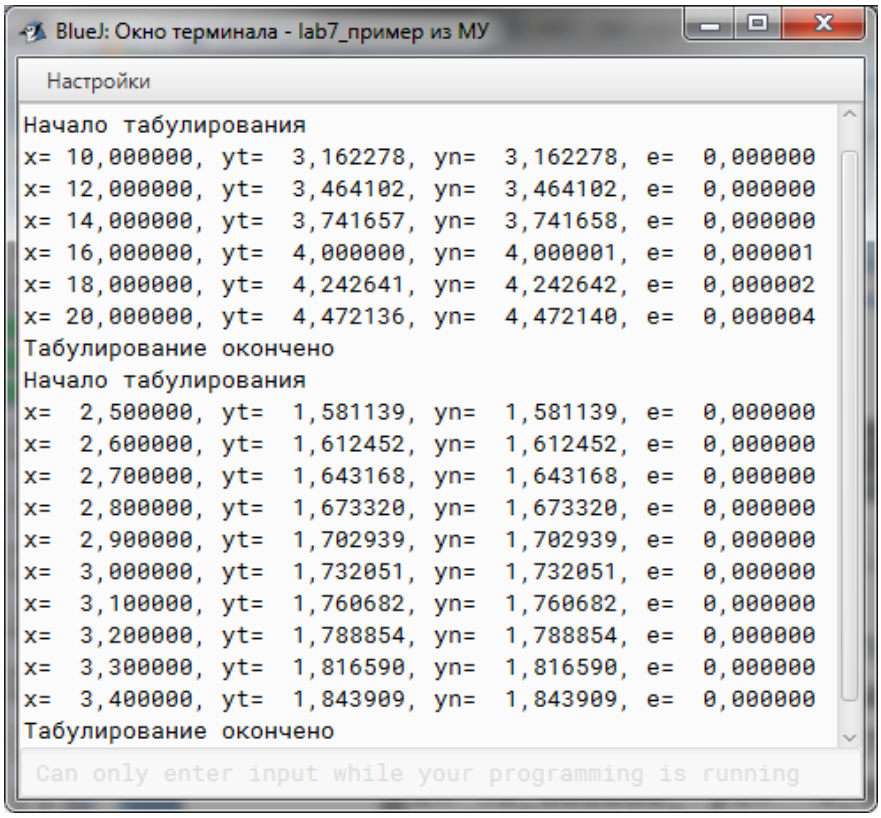


Рисунок 3.7 – Результаты работы программы

График функции, построенный по результатам табулирования функции на интервале [10, 100] с шагом 10 приведен на рисунке 3.8. График построен при помощи табличного процессора Excel.

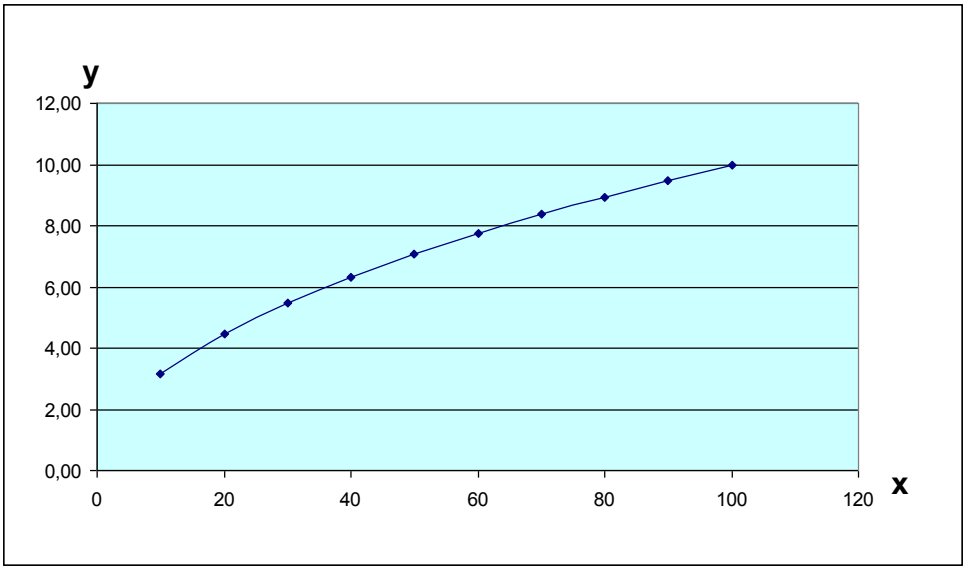


Рисунок 3.8 – График функции $y = \sqrt{x}$, построенный по результатам табулирования на интервале [10,100] с шагом 10

3.8. Контрольные вопросы

1) Опишите синтаксис и алгоритм работы оператора цикла с предусловием. Какое минимальное количество раз выполняется цикл с предусловием?

2) Опишите синтаксис и алгоритм работы оператора цикла с постусловием. Какое минимальное количество раз выполняется цикл с постусловием?

3) Опишите синтаксис и алгоритм работы оператора цикла с параметром (управляющей переменной).

4) Управляющие переменные каких типов можно использовать в цикле for языка Java?

5) От каких исходных данных зависит время выполнения составленных в лабораторной работе программ? Объясните почему.

6) От чего зависит точность вычисления значения функции по итерационной формуле?

7) Какое условие вы использовали для выхода из цикла табулирования функции? Почему мы никогда не попадаем в правую границу интервала при увеличении значения аргумента?

8) Можно ли использовать в качестве условия выхода из цикла результат сравнения на равенство значений вещественных переменных, полученных в результате выполнения операций над этими переменными?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ноутон П. Java 2: /П.Ноутон, Г.Шилдт. – СПб.: БХВ-Петербург, 2007. – 1072 с.

2. Яшкардин В. IEEE 754 – стандарт двоичной арифметики с плавающей точкой. – Электронный ресурс. – Режим доступа: http://www.softelectro.ru/ieee754.html.

3. Глушаков С. В. Программирование на Java 2: учеб. пособие для студ. вузов/ С. В. Глушаков. – Харьков: Фолио, 2003. – 544 с.