МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ»)

Факультет информационных систем и компьютерной безопасности

Кафедра искусственного интеллекта и цифровых технологий

КУРСОВОЙ ПРОЕКТ

по дисциплине: «Основы программирования и алгоритмизации»

Тема: «Программа анализа функции»

**Расчетно-пояснительная записка**

Разработал студент A.M.Ушаков

Подпись, дата Инициалы, фамилия

Руководитель О.В. Минакова

Подпись, дата Инициалы, фамилия

Нормоконтролер О.В. Минакова

Подпись, дата Инициалы, фамилия

Защищена Оценка

2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВА- ТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ВОРОНЕЖ- СКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ»)

Кафедра систем управления и информационных технологий в строительстве

ЗАДАНИЕ

на курсовой проект

по дисциплине Основы программирования и алгоритмизации

Тема: «Программа анализа функции»

Задание: F1=; F2= при x>1; F3 – обратный гиперболический косинус

Студент группы бИЦ-241 Ушаков Антон Михайлович

Фамилия, имя, отчество

Технические условия Windows 10, Microsoft Visual Studio 2022, язык программирования С

Содержание и объем проекта (графические работы, расчеты и прочее): 35 стр, 30 рисунков, 1 таблица. приложение

Сроки выполнения этапов:

Анализ и постановка задачи (10.09.24 – 05.10.24);

Разработка пошаговой детализации программы (06.10.24 – 11.11.24);

Реализация программы (11.11.24 – 05.12.24);

Тестирование программы (09.12.24 – 10.12.24);

Оформление пояснительной записки (12.12.24 – 17.12.24).

Срок защиты курсового проекта

Руководитель О.В. Минакова

Подпись, дата Инициалы, фамилия

Задание принял студент А.M. Ушаков

Подпись, дата Инициалы, фамилия

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 4](#_Toc187919286)

[**1. Постановка задачи** 5](#_Toc187919287)

[**2. Реализация программы** 9](#_Toc187919288)

[**3. Тестирование программы** 17](#_Toc187919289)

[**ЗАКЛЮЧЕНИЕ** 20](#_Toc187919290)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 21](#_Toc187919291)

[**ПРИЛОЖЕНИЕ А. Листинг программы** 22](#_Toc187919292)

# ВВЕДЕНИЕ

Анализ функций является одной из ключевых задач в математике и информатике, играющей важную роль в различных областях науки и техники. Распознавание и исследование свойств функций, таких как область определения, наличие экстремумов и поведение при различных значениях аргумента, имеют практическое применение в оптимизации, машинном обучении, экономике и многих других направлениях. Учитывая быстрое развитие технологий и потребности в более эффективных методах обработки данных, создание программного обеспечения для анализа функций становится особенно актуальным.

Цель данного курсового проекта заключается в разработке программы для анализа математических функций на языке программирования C. Данная программа позволит пользователям производить анализ функций, что значительно упростит решение задач в научных и инженерных областях.

В рамках проекта поставлены следующие задачи:

1. Изучить различные математические функции, определить их области допустимых значений (ОДЗ) и представить их в виде программного кода.

2. Разработать методы ввода и вывода данных, что обеспечит удобный интерфейс для пользователя.

3. Реализовать функции для вычисления производной, а также алгоритмы линейного поиска, поиска минимума и максимума функции, что позволит пользователю эффективно анализировать поведение функций.

4. Создать интерфейс для выбора нужной функции, что обеспечит легкость в использовании программы.

5. Провести тестирование программы для проверки корректности работы реализованных функций и алгоритмов.

1. **Постановка задачи**

Для начала определяются области определения функций. Для простоты функции обозначены F1, F2, F3 в порядке, соответствующем указанному в задании(слева направо).

F1 - график функции представлен на рисунке ниже.

Изображение выглядит как График, линия, Параллельный

Автоматически созданное описание

Рисунок 1 – график F1.

По графику можно понять, что функция определена при любых значениях x отличных от 0.

F2 – график функции на рисунке 2.

Изображение выглядит как линия, График, диаграмма, Параллельный

Автоматически созданное описание

Рисунок 2 -график F2.

Функция определена при любых значениях x.

F3 – график на рисунке 3.

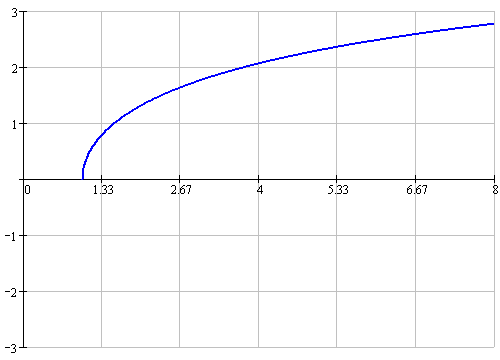


Рисунок 3 – график F3.

По графику определяется ОДЗ функции: .

Для организации управления необходимо реализовать диалог взаимодействия с пользователем. Программа должна предоставлять пользователю возможность выполнения следующих действий:

0) Выбор функции.

1) Вывод таблицы значений на экран и в файл;

2) Определение минимального и максимальное из вычисленных значений;

3) Линейного поиска значения аргумента при заданном значении функции (с определенной точностью);

4) Дифференцирование функции в заданной точке с заданной точностью (ошибкой приближения);

5) Задание режима и параметров расчета (интервал, шаг, кол-во значений);

6) Нахождение значения аргумента, при котором функция меняет знак.

7) Нахождения модуля разности между максимальным и минимальным значениями функции.

8) Проверка на пересечение графика функции с осью OY.

9) Завершение работы.

Для функций будет создана структура с массивами для аргумента и значения функций. Это упростит работу с ними и уменьшит размер кода.

Выбор функции реализован при помощи меню: пользователю будет предложен перечень доступных функций, находящихся под определённым номером.

Для выбора конкретной функции пользователь должен ввести соответствующее число.

Для 0 пункта через конструкцию “Switch” будет сделан выбор функции, с которой программа позже будет работать.

Для 1 пункта программа будет выводить информацию в консоль и сохранять её в файле. Информация будет находиться в массиве.

Для 2 пункта – программа будет использовать функции для нахождения минимума и максимума в массиве.

Для 3 пункта – программа будет искать ближайшее к заданному числу значение. Точность задается через #define.

Для 4 пункта – программа будет искать производную функции в определенной точке.

Для 5 пункта – выбор режима расчета через конструкцию “Switch”.

А) для заданного списком множества значений x (csv-файл);

Б) сгенерированных от заданного значения xmin с заданным шагом dx фиксированное кол-во значений (n=10);

В) случайным образом выбранных из заданного интервала (т.е. в заданном интервале (xmin; xmax).

Для 6 пункта – программа проверяет, имеют ли подряд идущие в массиве значения разные знаки

Для 7 пункта – нахождение модуля разности значений, полученных из 2 пункта.

Для 8 пункта – проверка, действительна ли функция при значении аргумента(x) = 0.

Для 9 пункта – завершение работы.

Контрольные примеры:

Берется значение для каждой функции, посчитанное другой программой(калькулятором). Для F1 – x = 1; для F2 – x=1 и 2. Получаются следующие значения: F1(1)=0.379; F2(1) =-1.104; F2(2)= 0.602; для F3 проверяется, равно ли число приблизительно тому, что мы ожидаем получить. Для x=4 y2.

Таблица 1 – Контрольные примеры для функций

|  |  |  |
| --- | --- | --- |
| **F1** | **F2** | **F3** |
| x=1 | x1=1; x2=2 | x=4 |
| y=0.379 | y1=-1.104; y2=0.602 | y=2 |

Для производной берется x=2. F2`(x).

Минимум и максимум функции F2 соответственно -1.104 и 0.602. Их разность по модулю- 1.706. Также по этим числам проверяется нахождение смены + на – в функции т.к. эти значения меняют знак.

Для линейного поиска ожидается, что он выведет корректный индекс. Это можно будет проверить без контрольного примера так как программа выводит значение функции и её индекс считается вручную.

Для вывода данных в файл ожидается соответствие значений в программе и в файле.

Для проверки пересечения с ОY ожидается ответ о наличии факта пересечения при F(0) ϵ **R** или отсутствии при других значениях F(0).

**2. Реализация программы**

Для использования дополнительных функций необходимо подключить к проекту соответствующие библиотеки с помощью директивы #include:

<stdio.h> – библиотека, содержащая стандартные функции, реализующие основные возможности ввода и вывода.

<stdlib.h> – библиотека, которая включает в себя функции, связанные с выделением памяти, управлением процессами, преобразованиями и другими, конкретно в данной программе нужна для генерации чисел.

<locale.h> – библиотека для локализации русского языка.

<math.h> – библиотека, содержащая функции для работы с математическими функциями.

Через #define определяется точность для вычисления ареа-косинуса и линейного поиска, количество повторов для разложения по рядам для ареа-косинуса и название файла для вывода.

После были созданы прототипы функций:

int input\_file(char\*);

int output\_to\_file(struct function values[], int);

double find\_max(struct function values[], int);

double find\_min(struct function values[], int);

unsigned long long factorial(int n);

int linear\_search(struct function values[], int, double);

double differentiate(double (\*func)(double), double, double);

void tabulation(int, struct function values[]);

void find\_sign\_changes(double (\*func)(double), struct function values[], int count);

int handle\_generated\_values\_mode(double, double);

int handle\_random\_values\_mode(double, double);

Для работы программы были созданы следующие функции:

“int input\_file(char\* inputfile)” – Принимает в качестве параметра inputfile указатель на строку, содержащую имя файла. В процессе выполнения функция открывает указанный файл, считывает его содержимое и сохраняет данные в соответствующие структуры. Возвращаемое значение функции указывает на успешность операции: -1 — в случае ошибки, что позволяет пользователю понять, была ли операция успешной. Название файла inputfile берется как строка, вводимая в функции “main”. Схема к данной функции на рисунке 4.

Изображение выглядит как диаграмма, текст, линия, Технический чертеж

Автоматически созданное описание

Рисунок 4 -Блок-схема input\_file.

“double differentiate(double (\*func)(double), double x, double h)” – принимает в качестве параметров указатель на функцию func, представляющую математическую функцию, x — точку, в которой необходимо вычислить производную, и h — малое приращение. В процессе выполнения функция использует определение производной, вычисляя отношение приращения функции к приращению аргумента. Возвращаемое значение функции представляет собой значение производной функции в точке x. Представляет из себя лишь 1 строку с return, в блок-схеме не нуждается.

“int linear\_search(struct function values[], int count, double target)” – принимает в качестве параметров указатель на массив значений в структуре values, представляющую набор значений, целое число count, указывающее количество элементов в массиве, и значение target, к которому будет осуществляться поиск. В процессе выполнения функция перебирает элементы массива и находит значение, наиболее близкое к target, после чего возвращает индекс этого значения в массиве. Если ближайшее значение не найдено, функция возвращает -1. Схема на рисунке 5.

Изображение выглядит как диаграмма, текст, линия, График

Автоматически созданное описание

Рисунок 5 – блок-схема linear\_search.

“void tabulation(int size, struct function values[])” – отображает значения функции в табличном формате, используя указатель на массив значений в структуре values. Параметр size определяет количество элементов в массиве. Каждый элемент массива представляет собой структуру, содержащую значения x и y. Функция сначала проверяет, является ли размер массива положительным. Затем она выводит заголовок таблицы, указывая колонки "x" и "y". После этого функция проходит по всем элементам массива и выводит значения x и y в формате, удобном для чтения. Функция не возвращает значения, так как ее основная задача — вывод данных на экран. Схема к функции представлена на рисунке 6.

Изображение выглядит как диаграмма, текст, Технический чертеж, План

Автоматически созданное описание

Рисунок 6 – блок-схема tabulation

“void find\_sign\_changes(double (\*func)(double), struct function values[], int count)” – предназначена для поиска и отображения значений аргумента, между которыми происходит изменение знака функции, заданной указателем на функцию func. Параметр values представляет указатель на массив значений в структуре, а параметр count указывает количество элементов в массиве. Функция перебирает значения аргумента и вычисляет результаты функции для последовательных пар значений. Если функция меняет знак между двумя последовательными значениями, функция выводит соответствующие аргументы на экран. Если изменений знака не обнаружено, выводится сообщение о том, что знак не меняется. Функция не возвращает значений, так как ее основная задача — вывод данных на экран. Блок-схема на рисунке 7.

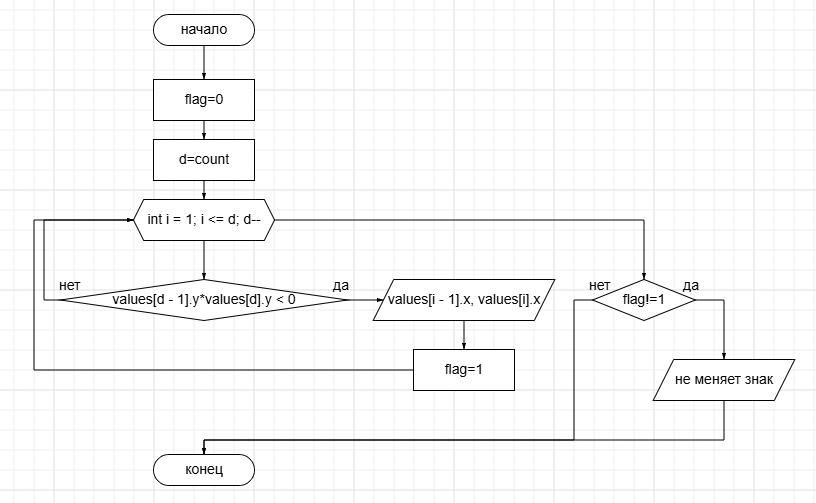


Рисунок 7 – блок-схема find\_sign\_changes.

“int output\_to\_file(struct function values[], int count)” – создает файл и записывает в него значения, хранящиеся в указателе на массив значений в структуре values. Параметр count задает количество элементов в массиве. В файле будут сохранены числа, которые представляют собой значения функций, соответствующие их аргументам и аргументы. Схема на рисунке 8.

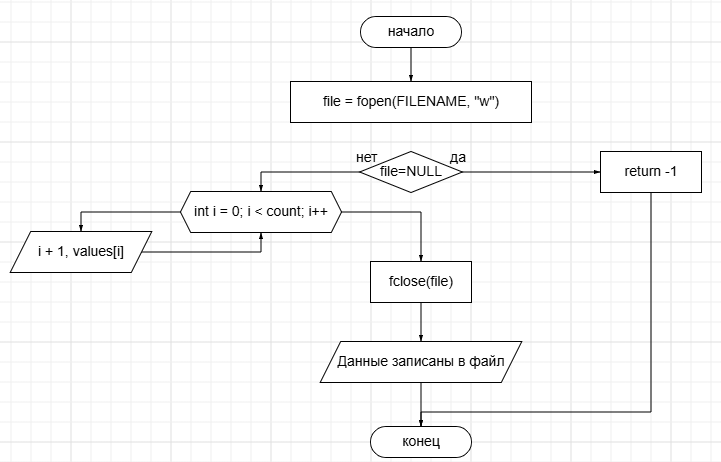


Рисунок 8 – блок-схема output\_to\_file.

“int handle\_generated\_values\_mode(double xmin, double dx)” – принимает заданный минимум xmin и шаг dx. Она добавляет в структуру значения функции и соответствующие аргументы на каждом шаге, начиная с xmin и увеличивая его на dx до достижения определенного предела. Блок-схема на рисунке 9.

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Автоматически созданное описание

Рисунок 9 – блок-схема handle\_generated\_values\_mode.

“int handle\_random\_values\_mode(double xmin, double xmax)” –  предназначена для генерации случайных значений функции в заданном диапазоне от xmin до xmax. Она будет добавлять в структуру значения функции и соответствующие аргументы, которые выбираются случайным образом в указанном диапазоне. Количество аргументов n задается в самой функии. Блок-схема на рисунке 10.

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Автоматически созданное описание

Рисунок 10 – блок-схема handle\_random\_values\_mode.

“double find\_max(struct function values[], int count)” – предназначена для поиска максимального значения values[].y в указателе на структуру values, где count — это количество элементов в массиве. Схема на рисунке 9.

Изображение выглядит как диаграмма, текст, линия, Шрифт

Автоматически созданное описание

Рисунок 11 – блок-схема find\_max.

“double find\_min(struct function values[], int count)” – предназначена для поиска минимального значения values[].y в указателе на структуру values, где count — это количество элементов в массиве. Аналогична функции выше, меняется только знак с < на >.

“unsigned long long factorial(int n)” – принимает целое число n и умножает его на значение result, изначально равное 1. После операций возвращает result. Используется в функции для поиска ареа-косинуса. Схема на рисунке 10.

Изображение выглядит как диаграмма, линия, текст, План

Автоматически созданное описание

Рисунок 12 – блок-схема factorial.

“double F1(double x)”; double F2(double x)”; double F3(double x)” – считают значения 1, 2 и 3(ареа-косинус) функции соответственно. Все 3 функции принимают значение x. Так как в 1 и 3 функции можно ввести значение x, при которой их не существует, умеренно добавлено возвращение NaN(не число) для таких значений. Схемы функций на рисунках ниже.

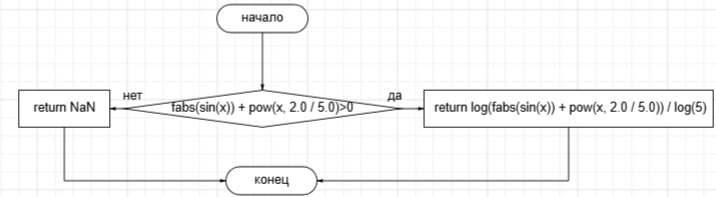




Рисунок 13 – блок-схема F1.

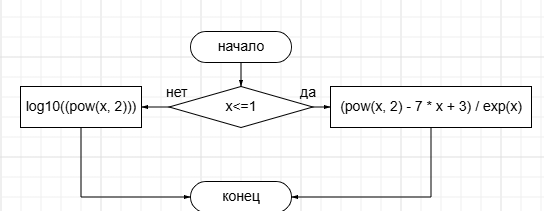


Рисунок 14 – блок-схема F2.

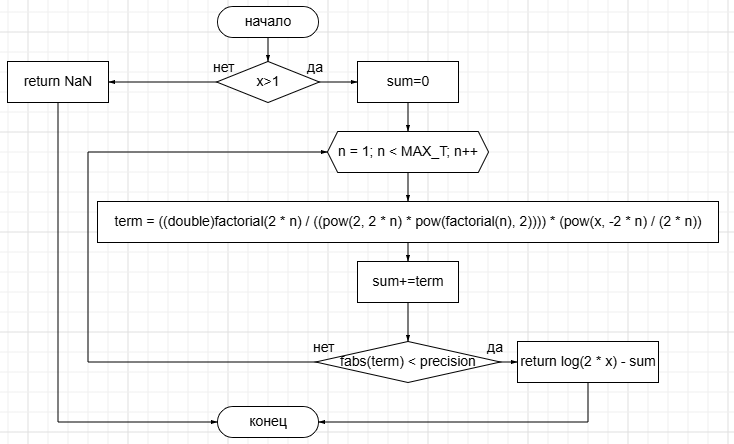


Рисунок 15 – блок-схема F3.

“int main()” – Объединяет все предыдущие функции для работы с ними. Именно в этой функции задаются значения, используемые в других функциях. Большая часть данной функции реализована через “Switch”. Это сделано для реализации меню. Схема функции на рисунках ниже.

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Автоматически созданное описание



Рисунок 16.1 – блок-схема main(case 1-4).

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Автоматически созданное описание

Рисунок 16.2 – блок-схема main(case 5-9 и default).

**3. Тестирование программы**

Тестирование начинается с 3 функции. Вводится значение x, не входящее в ОДЗ, например 0. Ожидаемо выводится nan. Для F1(0) результат аналогичен.



Рисунок 17 – результат для значений x вне ОДЗ.

Далее проверяются значения из контрольного примера. При F1(1); F2(1); F2(2); F3(4) получаются результаты, представленные на рисунках ниже.

Рисунок 18 – результаты для F1. Рисунок 19 – результаты для F2.



Рисунок 20 – результаты для F3.

Как можно заметить, числа соответствуют тем, что были представлены в контрольном примере. Для проверок F1 и F2 использовался файл “input.csv” с 2 числами в нём (1 и 2). Для F3 создавался другой файл с числом 4.

Проверим производную для F2(2). В программе приращение задается вручную и при крайне маленьких значениях приравнивается к 0. Поэтому возьмем приращение 0.00001.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 21– производная в точке.

Также нужно проверить нахождение минимума и максимума функций. Для F2: из значений 1 и 2 наибольший аргумент будет равен 0.602, наименьший – -1.104.



Рисунок 22 – минимум и максимум F2.

Теперь вводится несуществующее имя файла для варианта получения данных через файл. В таком случае программа примет это название и продолжит работу, но вывести значения не сможет.



Рисунок 23 – отсутствие значений при несуществующем файле.

Для линейного поиска по представленному на рисунке списку значений берется значение 0.37.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание 

Рисунок 24 – список значений Рисунок 25 – линейный поиск по значениям.

Получается индекс 1, что соответствует действительности. С этими же значениями функции проверяются данные, которые выводятся в файл.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 26 – вывод в файле.

Эти данные соответствуют тем, что выводит программа в консоль.

Для F2 при введенных значениях x 1 и 2 знак на индексе 0 отрицателен, на 1 – положителен. Программа выводит эту информацию корректно(рисунок 23)



Рисунок 27 – результат поиска смены знака.

Результат нахождения модуля разности также соответствует контрольному примеру.



Рисунок 28 – результат нахождения модуля разности

Для проверки на пересечение возьмем F1(0). При таком значении аргумента функция не существует, и программа выводит, что пересечения с ОY нет. Для F2(0) y=3, выводится сообщение о наличии пересечения с OY.

Рисунок 29 – результат для F1(0) Рисунок 30 – результат для F2(0)

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсового проекта была реализована программа, обеспечивающая работу с функциями. Программа способна выполнять следующие функции:

0) Выбор функции.

1) Вывод таблицы значений на экран и в файл;

2) Определение минимального и максимальное из вычисленных значений;

3) Линейного поиска значения аргумента при заданном значении функции (с определенной точностью);

4) Дифференцирование функции в заданной точке с заданной точностью (ошибкой приближения);

5) Задание режима и параметров расчета (интервал, шаг, кол-во значений);

6) Нахождение значения аргумента, при котором функция меняет знак.

7) Нахождения модуля разности между максимальным и минимальным значениями функции.

8) Проверка на пересечение графика функции с осью OY.

9) Завершение работы.

При выводе таблицы в каталоге проекта по выбору создается файл со значениями функции. Так, пользователь может переносить всю сохранённую информацию для работы в других программах.

Ссылка на репозиторий: <https://github.com/WhyMan55/KP>

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Минакова О. В., Курипта О. В. Практикум по Си: [Интернет-ресурс]. – URL: <https://sites.google.com/view/course-of-study1-c/> [Дата обращения: 29.12.2024].
2. Система вопросов и ответов Stack Overflow: [Интернет-ресурс]. – URL: <https://stackoverflow.com/> [Дата обращения: 28.12.2024].
3. Справочник по языку C: [Интернет-ресурс]. – URL: <https://learn.microsoft.com/ru-ru/cpp/c-language/> [Дата обращения: 18.12.2024].
4. Форум программистов CyberForum: [Интернет-ресурс]. – URL: <https://www.cyberforum.ru/c/> [Дата обращения: 17.12.2024].
5. Онлайн-редактор блок-схем: [Интернет-ресурс]. – URL: <https://programforyou.ru/block-diagram-redactor> [Дата обращения: 29.12.2024].
6. Бесплатный интерактивный курс по C: [Интернет-ресурс]. –URL: <https://www.learn-c.org/> [Дата обращения: 29.12.2024].
7. Брайан Керниган, Деннис Ритчи, Язык программирования Си: Учебное пособие, США: 1978 – 343 с. – ISBN 0-13-110163-3.
8. К.H. Кинг, C Programming: A Modern Approach: Книга, США: 1996 – 832 с. – ISBN-10: 0393979504; ISBN-13: 978-0393979503.
9. Солдатенко И. С. Основы программирования на языке Си: Учебное пособие. – Тверь: Твер. гос. ун-т, 2017. – 159 с. – ISBN 978-5-7609-1229.
10. Wikipedia: [Интернет-ресурс]. – URL:<https://wikipedia.org> [Дата обращения: 29.12.2024].

**ПРИЛОЖЕНИЕ А. Листинг программы**

#define \_CRT\_SECURE\_NO\_WARNINGS // Отключение предупреждений

#include <stdio.h> // Подключение стандартной библиотеки ввода/вывода

#include <stdlib.h> // Подключение стандартной библиотеки для работы с памятью и другими утилитами

#include <math.h> // Подключение математической библиотеки для работы с математическими функциями

#include <locale.h> // Подключение библиотеки для работы с локализацией

#define precision 0.015 // Определение точности для вычислений

#define FILENAME "output.csv" // Имя файла для вывода данных

#define MAX\_T 100 // Максимальное количество повторов для F3

// Определение структуры для хранения координат (x, y) функции

struct function {

double x; // Значение аргумента

double y; // Значение функции

};

int count = 0; // Глобальная переменная для хранения количества вычисленных значений

struct function\* values; // Указатель на массив значений функции

double F1(double); // Прототип функции F1

double F2(double); // Прототип функции F2

double F3(double); // Прототип функции F3

double (\*func)(double) = F1; // Указатель на функцию, по умолчанию указывает на F1

// Прототипы функций для работы с данными

int input\_file(char\*); // Функция для чтения данных из файла

int output\_to\_file(struct function values[], int); // Функция для записи данных в файл

double find\_max(struct function values[], int); // Функция для нахождения максимального значения

double find\_min(struct function values[], int); // Функция для нахождения минимального значения

unsigned long long factorial(int n); // Функция для вычисления факториала

int linear\_search(struct function values[], int, double); // Функция для линейного поиска

double differentiate(double (\*func)(double), double, double); // Функция для дифференцирования

void tabulation(int, struct function values[]); // Функция для табуляции значений

void find\_sign\_changes(double (\*func)(double), struct function values[], int count); // Функция для нахождения изменений знака функции

int handle\_generated\_values\_mode(double, double); // Функция для обработки сгенерированных значений

int handle\_random\_values\_mode(double, double); // Функция для обработки случайных значений

int main() {

setlocale(LC\_CTYPE, "RUS"); // Установка русской локали для корректного отображения текста

int mode; // Переменная для хранения выбранного режима работы

double xmin, xmax, dx; // Переменные для хранения границ и шага

while (1) { // Бесконечный цикл для меню

// Вывод меню на экран

printf("\nМеню:\n");

printf("0) Выбор функции для работы(по умолчанию F1)\n");

printf("1) Вывод таблицы значений на экран и в файл\n");

printf("2) Определение минимального и максимального из вычисленных значений\n");

printf("3) Линейный поиск значения аргумента при заданном значении функции\n");

printf("4) Дифференцирование функции в заданной точке\n");

printf("5) Задание режима и параметров расчета\n");

printf("6) Определение изменения знака функции\n");

printf("7) Модуль разности минимума/максимума функции.\n");

printf("8) Проверка пересечения OY.\n");

printf("9) Завершение работы\n");

printf("Выберите опцию: ");

scanf("%d", &mode); // Считывание выбранной опции

switch (mode) { // Обработка выбранной опции

case 0: { // Выбор функции

int choice\_func; // Переменная для хранения выбора функции

printf("Выберите функцию(F1 - 1,F2 - 2,F3 - 3):\n");

scanf("%d", &choice\_func); // Считывание выбора функции

switch (choice\_func) { // Установка выбранной функции

case 1: { func = F1; break; }

case 2: { func = F2; break; }

case 3: { func = F3; break; }

default: func = F1; break; // По умолчанию устанавливаем F1

}

break;

case 1: // Вывод таблицы значений

tabulation(count, values); // Вызов функции для табуляции значений

break;

case 2: // Определение минимального и максимального значений

if (count == 0) { // Проверка, были ли вычислены значения

printf("Нет вычисленных значений.\n"); // Сообщение об отсутствии значений

}

else {

// Вывод минимального и максимального значений

printf("Минимальное значение: %lf\n", find\_min(values, count));

printf("Максимальное значение: %lf\n", find\_max(values, count));

}

break;

case 3: { // Линейный поиск значения функции

double target; // Переменная для хранения искомого значения

printf("Введите значение функции для поиска: ");

scanf("%lf", &target); // Считывание искомого значения

int index = linear\_search(values, count, target); // Поиск значения

if (index != -1) { // Если значение найдено

printf("Значение найдено на индексе: %d\n", index);

}

else {

printf("Значение не найдено.\n"); // Сообщение о том, что значение не найдено

}

break;

}

case 4: { // Дифференцирование функции

double x, h; // Переменные для точки и приращения

printf("Введите точку для дифференцирования: ");

scanf("%lf", &x); // Считывание точки

printf("Введите приращение h: ");

scanf("%lf", &h); // Считывание приращения

double derivative = differentiate(func, x, h); // Вычисление производной

printf("Производная в точке %lf: %lf\n", x, derivative); // Вывод результата

break;

}

case 5: { // Выбор режима и параметров расчета

int choice; // Переменная для хранения выбора режима

printf("Выберите режим:\n");

printf("1) Заданные значения (cvs-файл)\n");

printf("2) Сгенерированные от xmin с шагом dx(n=10)\n");

printf("3) Случайные значения из интервала (xmin; xmax)\n");

printf("Выберите режим: ");

scanf("%d", &choice); // Считывание выбора режима

switch (choice) { // Обработка выбора режима

case 1: { // Заданные значения из файла

printf("Введите имя файла\n");

char inputfile[255]; // Массив для хранения имени файла

scanf("%s", inputfile); // Считывание имени файла

input\_file(inputfile); // Вызов функции для чтения данных из файла

for (int i = 0; i < count; i++) // Вычисление значений функции для считанных x

values[i].y = func(values[i].x);

break;

}

case 2: { // Сгенерированные значения

printf("Введите xmin: ");

scanf("%lf", &xmin); // Считывание xmin

printf("Введите шаг dx: ");

scanf("%lf", &dx); // Считывание шага

handle\_generated\_values\_mode(xmin, dx); // Вызов функции для обработки сгенерированных значений

break;

}

case 3: { // Случайные значения

printf("Введите xmin: ");

scanf("%lf", &xmin); // Считывание xmin

printf("Введите xmax: ");

scanf("%lf", &xmax); // Считывание xmax

handle\_random\_values\_mode(xmin, xmax); // Вызов функции для обработки случайных значений

break;

}

default:

printf("Неверный выбор. Попробуйте снова.\n"); // Сообщение об ошибке выбора

}

break;

}

case 6: { // Определение изменения знака функции

find\_sign\_changes(func, values, count); // Вызов функции для нахождения изменений знака

break;

}

case 7: // Вычисление модуля разности максимума и минимума

printf("Модуль разности макс/мин - %lg \n", fabs(find\_max(values, count) - find\_min(values, count))); // Вывод результата

break;

case 8: // Проверка пересечения с осью OY

if (isnan(func(0)) || isinf(func(0))) { // Проверка, есть ли значение функции в точке x=0

printf("Функция не пересекает OY.\n"); // Сообщение о том, что функция не определена

}

else {

printf("Функция пересекает OY\n"); // Вывод сообщения о том, что функция пересекает OY

}

break;

case 9: // Завершение работы программы

printf("Завершение работы.\n"); // Сообщение о завершении работы

free(values); // Освобождение выделенной памяти

return 0; // Выход из программы

default: // Обработка неверного выбора

printf("Неверный выбор. Попробуйте снова.\n"); // Сообщение об ошибке выбора

break;

}

}

}

}

// Функция для табуляции значений функции

void tabulation(int size, struct function values[]) {

if (count == 0) { // Проверка, есть ли вычисленные значения

printf("Нет вычисленных значений.\n"); // Сообщение об отсутствии значений

}

else {

printf("| x | y |\n"); // Заголовок таблицы

for (int i = 0; i < count; i++) { // Цикл по всем значениям

printf("| %11.4lf | %11.4lf |\n", values[i].x, values[i].y); // Вывод значений x и y

}

int a;

printf("Вывести в файл?(1 - да; остальные - нет)"); // Запрос на вывод в файл

scanf("%d", &a); // Считывание выбора пользователя

if (a == 1) {

output\_to\_file(values, count); // Вызов функции для записи в файл

}

}

}

// Функция для вычисления производной функции в заданной точке

double differentiate(double (\*func)(double), double x, double h) {

return (func(x + h) - func(x)) / h; // Формула для численного дифференцирования

}

// Линейный поиск значения в массиве

int linear\_search(struct function values[], int count, double target) {

for (int i = 0; i < count; i++) { // Цикл по всем значениям

if (fabs(values[i].y - target) < precision) { // Проверка на близость значения

return i; // Возвращаем индекс найденного значения

}

}

return -1; // Если значение не найдено

}

// Функция для считывания данных из файла

int input\_file(char\* inputfile) {

FILE\* file = fopen(inputfile, "r"); // Открытие файла для чтения

if (file == NULL) {

return -1; // Ошибка при открытии файла

}

values = (struct function\*)malloc(MAX\_T \* sizeof(struct function)); // Выделение памяти для значений

count = 0; // Сброс счетчика

while (fscanf(file, "%lf", &values[count].x) != EOF) { // Считывание значений x из файла

values[count].y = 0; // Инициализация y

count++; // Увеличение счетчика

}

fclose(file); // Закрытие файла

return count; // Возвращаем количество считанных значений

}

// Функция для записи значений в файл

int output\_to\_file(struct function values[], int count) {

FILE\* file = fopen(FILENAME, "w"); // Открытие файла для записи

if (file == NULL) {

return -1; // Ошибка при открытии файла

}

for (int i = 0; i < count; i++) { // Цикл по всем значениям

fprintf(file, "%d, %lf, %lf\n", i + 1, values[i].x, values[i].y); // Запись значений в файл

}

fclose(file); // Закрытие файла

printf("Данные записаны в файл %s\n", FILENAME); // Сообщение о записи

return 0; // Успешное завершение

}

// Функция для нахождения изменений знака функции

void find\_sign\_changes(double (\*func)(double), struct function values[], int count) {

int flag = 0; // Флаг для отслеживания изменений знака

int d = count; // Сохраняем значение count

for (int i = 1; i < d; d--) { // Цикл по значениям

if (values[d - 1].y \* values[d].y < 0) { // Проверка на изменение знака

printf("Между x = %lf и x = %lf\n", values[i - 1].x, values[i].x); // Вывод интервала

flag = 1; // Установка флага

}

}

if (flag != 1) // Если не было изменений знака

printf("Не меняет знак в заданном интервале.\n"); // Сообщение об отсутствии изменений

}

// Функция для нахождения максимального значения

double find\_max(struct function values[], int count) {

double max = values[0].y; // Инициализация максимального значения

for (int i = 1; i < count; i++) { // Цикл по всем значениям

if (values[i].y > max) max = values[i].y; // Обновление максимума

}

return max; // Возврат максимального значения

}

// Функция для нахождения минимального значения

double find\_min(struct function values[], int count) {

double min = values[0].y; // Инициализация минимального значения

for (int i = 1; i < count; i++) { // Цикл по всем значениям

if (values[i].y < min) min = values[i].y; // Обновление минимума

}

return min; // Возврат минимального значения

}

// Обработка режима генерации значений

int handle\_generated\_values\_mode(double xmin, double dx) {

int n = 10; // Количество значений

count = 0; // Сброс счетчика

values = (struct function\*)malloc(n \* sizeof(struct function)); // Выделение памяти

for (double x = xmin; count < n; x += dx) { // Генерация значений

values[count].x = x; // Присвоение x

values[count].y = func(x); // Вычисление y

count++; // Увеличение счетчика

}

return count; // Возврат количества значений

}

// Обработка режима случайных значений

int handle\_random\_values\_mode(double xmin, double xmax) {

int n; // Количество значений

printf("Сколько значений? "); // Запрос на количество

scanf("%d", &n); // Считывание количества

values = (struct function\*)malloc(n \* sizeof(struct function)); // Выделение памяти

count = 0; // Сброс счетчика

while (count < n) { // Генерация случайных значений

double x = xmin + 1.f \* (xmax - xmin) \* rand() / RAND\_MAX; // Генерация случайного x

values[count].x = x; // Присвоение x

values[count].y = func(x); // Вычисление y

count++; // Увеличение счетчика

}

return count; // Возврат количества значений

}

// Функция для вычисления факториала

unsigned long long factorial(int n) {

if (n == 0 || n == 1) return 1; // Базовый случай

unsigned long long result = 1; // Инициализация результата

for (int i = 2; i <= n; i++) { // Цикл для вычисления факториала

result \*= i; // Умножение

}

return result; // Возврат факториала

}

double F1(double x) {

if (fabs(sin(x)) + pow(x, 2.0 / 5.0) > 0) // Проверка на положительность

return log(fabs(sin(x)) + pow(x, 2.0 / 5.0)) / log(5); // Возврат логарифма

else return NAN; // Возврат NaN в случае ошибки

}

double F2(double x) {

if (x <= 1) {

return (pow(x, 2) - 7 \* x + 3) / exp(x); // Вычисление для x <= 1

}

else {

return log10((pow(x, 2))); // Вычисление для x > 1

}

}

double F3(double x) {

if (x > 1) {

double sum = 0.0; // Инициализация суммы

for (int n = 1; n < MAX\_T; n++) { // Цикл для вычисления суммы

double term = ((double)factorial(2 \* n) / ((pow(2, 2 \* n) \* pow(factorial(n), 2)))) \* (pow(x, -2 \* n) / (2 \* n)); //расчёт 1 члена

sum += term; // Добавление члена к сумме

if (fabs(term) < precision) { // Проверка на точность

break; // Прерывание, если достигнута необходимая точность

}

}

return log(2 \* x) - sum; // Вычисление функции

}

else return NAN; // Возврат NaN в случае ошибки

}