Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Вычисление математических функций с использованием рядов»**

**Выполнил**:

студентка группы 3824Б1ПМ1

Фирсова А.М

**Проверила**:

Бусько П.В.

Нижний Новгород

2024

**Содержание**

[Введение 3](#_Toc193208802)

[Постановка задачи 4](#_Toc193208803)

[Руководство пользователя 5](#_Toc193208804)

[Описание программной реализации 7](#_Toc193208805)

[Результаты экспериментов 17](#_Toc193208806)

[Заключение 26](#_Toc193208807)

[Литература 27](#_Toc193208808)

[Приложение 28](#_Toc193208809)

# Введение

В математике и вычислительной технике часто возникает задача вычисления значений различных функций, таких как тригонометрические, экспоненциальные и логарифмические функции. Однако точное вычисление этих функций может быть сложным или даже невозможным в некоторых случаях, особенно если речь идет о ручных вычислениях или ограниченных вычислительных ресурсах. Для решения этой проблемы широко используются приближенные методы, одним из которых является разложение функций в ряды Тейлора.

Ряд Тейлора — это бесконечная сумма членов, которая позволяет аппроксимировать функцию в окрестности некоторой точки. Используя конечное количество членов ряда, можно получить приближенное значение функции с заданной точностью. Этот метод особенно полезен в вычислительной математике, где точность и скорость вычислений играют ключевую роль.

# Постановка задачи

В рамках данной лабораторной работы необходимо:

1. Требуется реализовать на языке программирования C подсчёт значений четырёх математических функций:

2. Реализовать два подхода для вычисления сумм рядов:

* Прямой порядок: вычисление суммы ряда от первого члена к последнему.
* Обратный порядок: вычисление суммы ряда от последнего члена к первому.

3. Исследовать влияние количества итераций на точность вычислений. Для этого:

* Вычислить значение функции с использованием рядов Тейлора для заданного количества итераций.
* Сравнить полученное значение с точным значением функции, вычисленным с помощью стандартных математических функций.
* Оценить погрешность вычислений на каждом шаге.

# Руководство пользователя

**Описание работы программы.**

Программа предназначена для вычисления значений функций с помощью ряда Тейлора. Кроме того, программа вычисляет погрешность на каждом шаге итерации, сравнивая результат с точным значением функции, полученным с помощью стандартных математических функций.

**Действия пользователя:**

1. Запуск программы:

Запустите программу. На экране появится меню с выбором функций.

1. Выбор функции:

Введите номер функции, которую хотите вычислить (1 для синуса, 2 для косинуса, 3 для экспоненты, 4 для натурального логарифма).

Если вы хотите выйти из программы, введите 5.

1. Ввод значения x:

После выбора функции программа запросит ввести значение x. Введите число в диапазоне от 0.5 до 1.

1. Ввод количества итераций:

Программа запросит ввести количество итераций. Введите целое положительное число (например, 10, 20 и т.д.).

1. Получение результатов:

Программа вычислит значение функции с использованием рядов Тейлора для прямого и обратного порядка суммирования.

1. На экране появятся результаты:

* Значение функции, вычисленное с использованием рядов Тейлора.
* Погрешность на каждом шаге итерации.

1. Повторение вычислений:

После вывода результатов программа вернется в главное меню. Вы можете выбрать другую функцию или выйти из программы.

1. Выход из программы:

Чтобы выйти из программы, выберите пункт 5 в главном меню.

Пример вывода в консоль(*рис.1*):

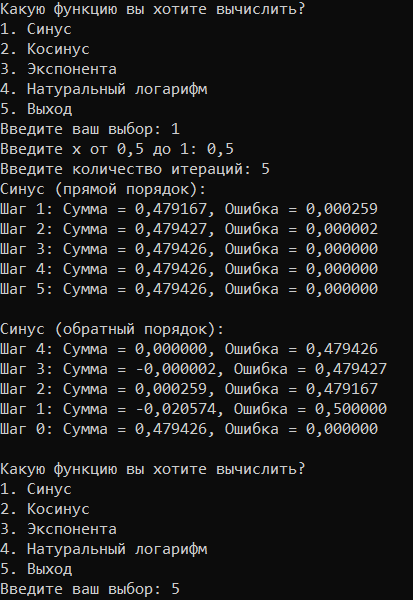


Рис. 1. Пример вывода в консоль

# Описание программной реализации

1. Файл ‘laboratory\_work.h’.

В нем находятся заголовочные файлы (*рис.2*), структура (*рис.3*) и прототипы функций (*рис.4*).

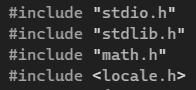


Рис. 2. Заголовочные файлы

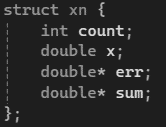


Рис. 3. Структура

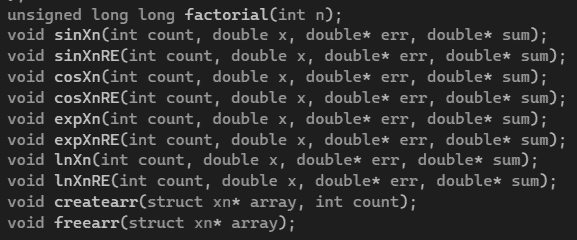


Рис. 4. Прототипы функций

Подключенные библиотеки:

* “stdio.h”, используется для ввода и вывода данных в консольном приложении;
* “stdlib.h”, используется для работы с динамической памятью;
* “math.h”, предоставляет набор математических функций и констант, которые используются для выполнения различных математических операций;
* <locale.h>, позволяет подключить русский язык;

Описание структуры:

В программе используется структура xn, которая предназначена для хранения данных, связанных с вычислением значений функций с использованием рядов Тейлора.

Поля структуры:

* int count:
  + Назначение: Хранит количество итераций (количество членов ряда, используемых для вычисления).
  + Тип: Целое число (int).
  + Использование: Определяет размер массивов err и sum, а также количество шагов в вычислениях.
* double x:
  + Назначение: Хранит значение аргумента x, для которого вычисляется функция.
  + Тип: Вещественное число (double).
  + Использование: Передается в функции для вычисления значений функций (например, , , ,).
* double\* err:
  + Назначение: Указатель на массив, в котором хранятся значения погрешностей на каждом шаге итерации.
  + Тип: Указатель на double.
  + Использование: Массив выделяется динамически с помощью malloc в функции createarr. В этом массиве сохраняется разница между точным значением функции и приближенным значением, вычисленным с помощью ряда Тейлора.
* double\* sum:
  + Назначение: Указатель на массив, в котором хранятся промежуточные суммы ряда на каждом шаге итерации.
  + Тип: Указатель на double.
  + Использование: Массив выделяется динамически с помощью malloc в функции createarr. В этом массиве сохраняются частичные суммы ряда Тейлора на каждом шаге.

Прототипы функций:

* unsigned long long factorial(int n), в аргумент принимает int (число для вычисления факториала), возвращает unsigned long long.
* void sinXn(int count, double x, double\* err, double\* sum), в аргумент принимает int (количество итераций), double (значение аргумента функции), указатель на массив double, указатель на массив double.
* void sinXnRE(int count, double x, double\* err, double\* sum), в аргумент принимает int (количество итераций), double (значение аргумента функции), указатель на массив double, указатель на массив double.
* void cosXn(int count, double x, double\* err, double\* sum), в аргумент принимает int (количество итераций), double (значение аргумента функции), указатель на массив double, указатель на массив double.
* void cosXnRE(int count, double x, double\* err, double\* sum), в аргумент принимает int (количество итераций), double (значение аргумента функции), указатель на массив double, указатель на массив double.
* void expXn(int count, double x, double\* err, double\* sum), в аргумент принимает int (количество итераций), double (значение аргумента функции), указатель на массив double, указатель на массив double.
* void expXnRE(int count, double x, double\* err, double\* sum), в аргумент принимает int (количество итераций), double (значение аргумента функции), указатель на массив double, указатель на массив double.
* void lnXn(int count, double x, double\* err, double\* sum), в аргумент принимает int (количество итераций), double (значение аргумента функции), указатель на массив double, указатель на массив double.
* void lnXnRE(int count, double x, double\* err, double\* sum), в аргумент принимает int (количество итераций), double (значение аргумента функции), указатель на массив double, указатель на массив double.
* void createarr(struct xn\* array, int count), в аргумент принимает указатель на структуру, int(количество итераций).
* void freearr(struct xn\* array), в аргумент принимает указатель на структуру.

2. Файл ‘Function.cpp’.

* Функция “factorial” (*рис.5*)

Функция вычисляет факториал числа.

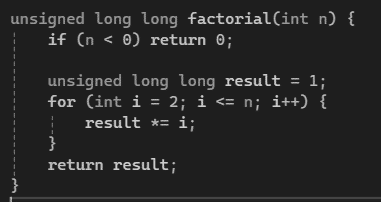


Рис. 5. Функция "factorial"

* Функция “sinXn” (*рис.6*)
* Назначение: Вычисляет значение синуса (sin(x)) с использованием ряда Тейлора в прямом порядке (от первого члена к последнему).
* Что делает:

Принимает количество итераций count, значение x, указатели на массивы err и sum.

Вычисляет сумму ряда Тейлора для синуса на каждом шаге.

Сохраняет промежуточные суммы в массив sum.

Вычисляет погрешность на каждом шаге и сохраняет её в массив err.

Выводит результаты на каждом шаге.

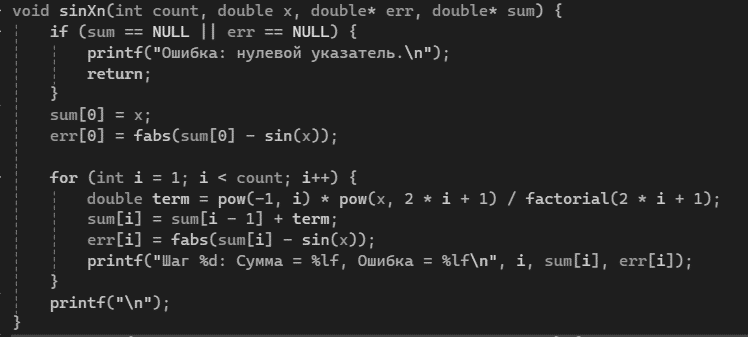


Рис. 6. Функция "sinXn"

* Функция “sinXnRE” (*рис.7*)
* Назначение: Вычисляет значение синуса (sin(x)) с использованием ряда Тейлора в обратном порядке (от последнего члена к первому).
* Что делает:

Аналогична функции sinXn, но суммирование начинается с последнего члена ряда.

Используется для сравнения точности вычислений при прямом и обратном порядке суммирования.

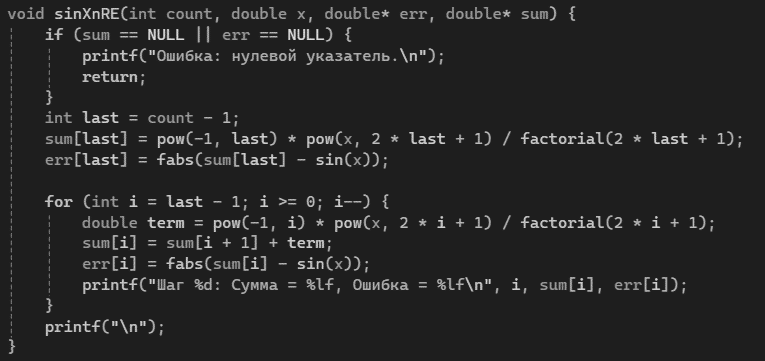


Рис. 7. Функция "sinXnRE"

* Функция “cosXn” (*рис.8*)
* Назначение: Вычисляет значение косинуса (cos(x)) с использованием ряда Тейлора в прямом порядке (от первого члена к последнему).
* Что делает:

Аналогично функции sinXn

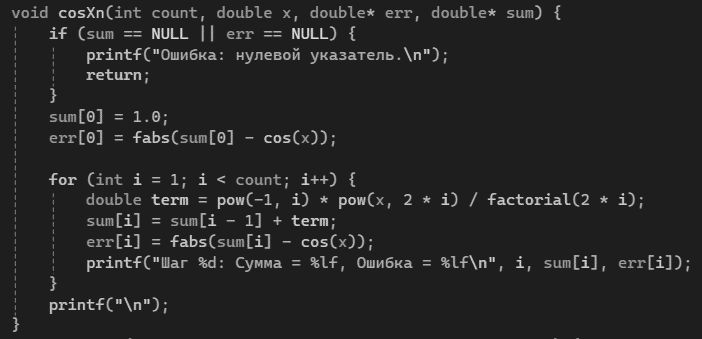


Рис. 8. Функция "cosXn"

* Функция “cosXnRE” (*рис.9*)
* Назначение: Вычисляет значение косинуса (cos(x)) с использованием ряда Тейлора в обратном порядке (от последнего члена к первому).
* Что делает:

Аналогична функции sinXnRE.

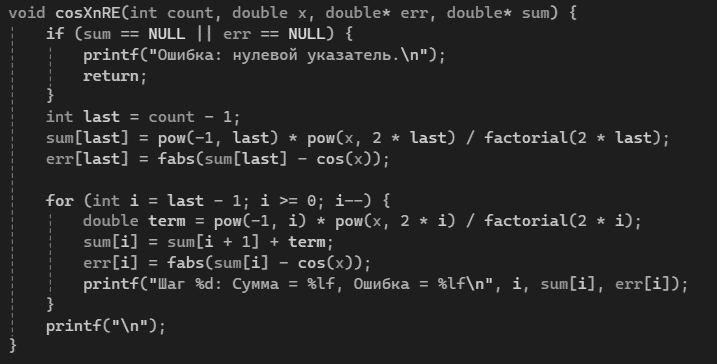


Рис. 9. Функция "cosXnRE"

* Функция “expXn” (*рис.10*)
* Назначение: Вычисляет значение экспоненты (exp(x)) с использованием ряда Тейлора в прямом порядке (от первого члена к последнему).
* Что делает:

Аналогично функции sinXn

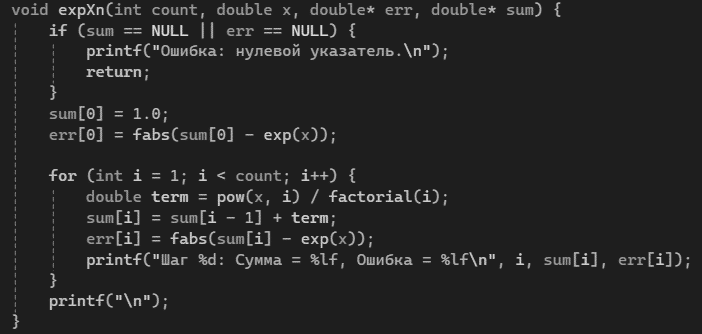


Рис. 10. Функция "expXn"

* Функция “expXnRE” (*рис.11*)
* Назначение: Вычисляет значение экспоненты (exp(x)) с использованием ряда Тейлора в обратном порядке (от последнего члена к первому).
* Что делает:

Аналогична функции sinXnRE.

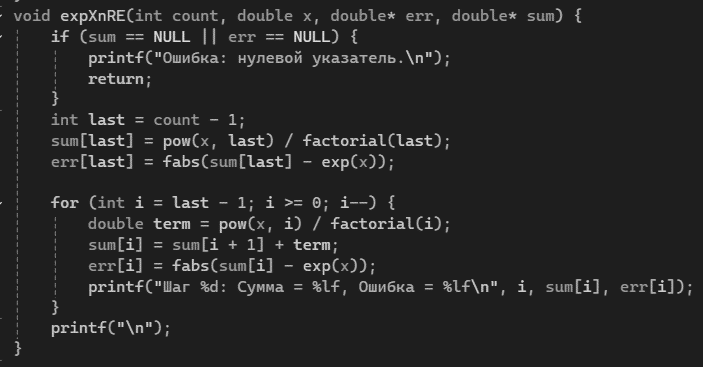


Рис. 11. Функция "expXnRE"

* Функция “lnXn” (*рис.12*)
* Назначение: Вычисляет значение экспоненты (ln(1+x)) с использованием ряда Тейлора в прямом порядке (от первого члена к последнему).
* Что делает:

Аналогично функции sinXn

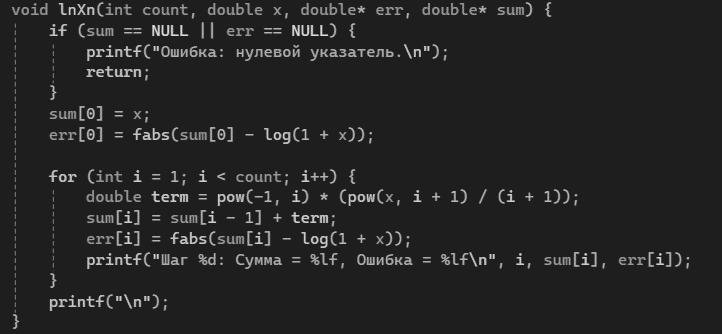


Рис. 12.. Функция "lnXn"

* Функция “lnXnRE” (*рис.13*)
* Назначение: Вычисляет значение экспоненты (ln(1+x)) с использованием ряда Тейлора в обратном порядке (от последнего члена к первому).
* Что делает:

Аналогична функции sinXnRE.

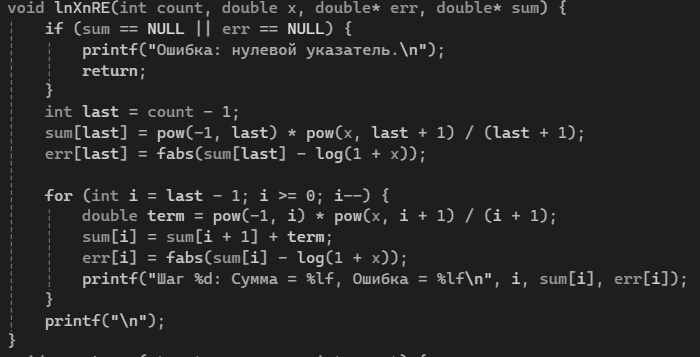


Рис. 13. Функция "lnXnRE"

* Функция “createarr” (*рис.14*)
* Назначение: Выделяет память для массивов sum и err в структуре xn.
* Что делает:

Принимает указатель на структуру xn и количество итераций count.

Выделяет память для массивов sum и err размером count.

Инициализирует массивы нулевыми значениями.

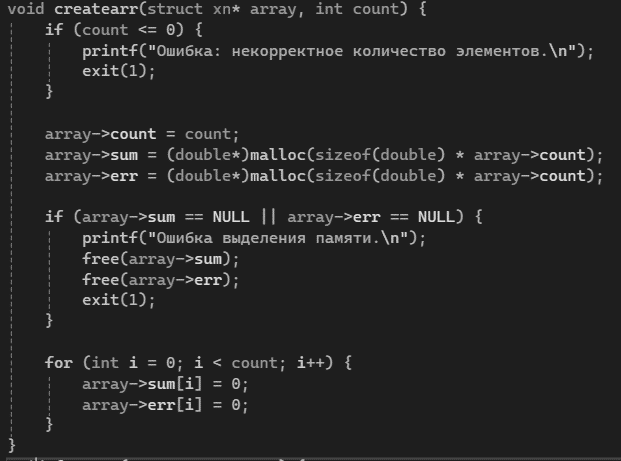


Рис. 14. Функция "creatarr"

* Функция “freearr” (*рис.15*)
* Назначение: Освобождает память, выделенную для массивов sum и err в структуре xn.
* Что делает:

Принимает указатель на структуру xn.

Освобождает память, выделенную для массивов sum и err.

Устанавливает указатели в NULL и обнуляет значение count.

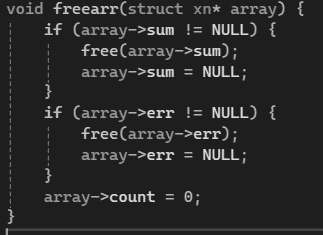


Рис. 15. Функция "freearr"

# Результаты экспериментов

Для проведения эксперимента рассмотрим погрешность между нашей и библиотечной реализациями значений функций в конкретной точке. Для функций возьмем . Я рассмотрела все необходимые способы суммирования с разным числом слагаемых.

Все данные представлены в таблицах и на графиках. Ось абсцисс отвечает за число слагаемых, а ось ординат – за погрешность.

**1. Прямое суммирование.**

1. Функция sin(x) (*таблица 1, график 1*).

Таблица 1

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,008138 |
| 2 | 0,000196 |
| 3 | 0,000003 |
| 4 | 0,000000 |
| 5 | 0,000000 |
| 6 | 0,000000 |
| 7 | 0,000000 |
| 8 | 0,000000 |
| 9 | 0,000000 |
| 10 | 0,000000 |

График 1

На *Графике 1* представлен график зависимости погрешности от числа слагаемых для функции sin(x). Видно, что уже при 3 слагаемых точность очень высока и практически не отличается от “эталонной”.

2) Функция cos(x) (*таблица 2, график 2*).

Таблица 2

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,040302 |
| 2 | 0,001364 |
| 3 | 0,000025 |
| 4 | 0,000000 |
| 5 | 0,000000 |
| 6 | 0,000000 |
| 7 | 0,000000 |
| 8 | 0,000000 |
| 9 | 0,000000 |
| 10 | 0,000000 |

График 2

Также, как и с синусом, ряд с косинусом очень быстро сходится (см. График 2).

3) Функция (*таблица 3, график 3*).

Таблица 3

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,718282 |
| 2 | 0,218282 |
| 3 | 0,051615 |
| 4 | 0,009948 |
| 5 | 0,001615 |
| 6 | 0,000226 |
| 7 | 0,000028 |
| 8 | 0,000003 |
| 9 | 0,000000 |
| 10 | 0,000000 |

График 3

С экспонентой ситуация несколько иная. *На Графике 3* видно, что кривая дольше прижимается к оси абсцисс, по сравнению с двумя другими функциями, которые мы уже рассмотрели.

4) Функция (*таблица 4, график 4*).

Таблица 4

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,193147 |
| 2 | 0,140186 |
| 3 | 0,109814 |
| 4 | 0,090186 |
| 5 | 0,076481 |
| 6 | 0,066377 |
| 7 | 0,058623 |
| 8 | 0,052488 |
| 9 | 0,047512 |
| 10 | 0,043397 |

График 4

График для логарифма также плавно опускается (см. *График 4*). Точность достаточно высокая.

**2. Обратное суммирование.**

1. Функция sin(x) (*таблица 5, график 5*).

Таблица 5

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,841471 |
| 2 | 0,841471 |
| 3 | 0,841471 |
| 4 | 0,841471 |
| 5 | 0,841471 |
| 6 | 0,841468 |
| 7 | 0,841667 |
| 8 | 0,833333 |
| 9 | 1,000000 |
| 10 | 0,000000 |

График 5

На *Графике 5* видно, что при 10 слагаемых значение ряда абсолютно не отличается от истинного значения.

2) Функция cos(x) (*таблица 6, график 6*).

Таблица 6

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,540302 |
| 2 | 0,540302 |
| 3 | 0,540302 |
| 4 | 0,540302 |
| 5 | 0,540303 |
| 6 | 0,540278 |
| 7 | 0,541667 |
| 8 | 0,500000 |
| 9 | 1,000000 |
| 10 | 0,000000 |

График 6

Точность такая же высокая, ряд с 8 слагаемыми недалек от истины (см. *График 6*).

3) Функция (*таблица 7, график 7*).

Таблица 7

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 2,718279 |
| 2 | 2,718254 |
| 3 | 2,718056 |
| 4 | 2,716667 |
| 5 | 2,708333 |
| 6 | 2,666667 |
| 7 | 2,500000 |
| 8 | 2,000000 |
| 9 | 1,000000 |
| 10 | 0,000000 |

График 7

Плавное уменьшение сохраняется, при 10 слагаемых значение истинно.

4) Функция (*таблица 8, график 8*).

Таблица 8

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,702238 |
| 2 | 0,591127 |
| 3 | 0,716127 |
| 4 | 0,573270 |
| 5 | 0,739937 |
| 6 | 0,539937 |
| 7 | 0,789937 |
| 8 | 0,456603 |
| 9 | 0,956603 |
| 10 | 0,043397 |

График 8

Погрешность колеблется, но к 10 слагаемому значение близко к истинному.

# Заключение

В ходе лабораторной работы был реализован подсчет необходимых математических функций через ряды Тейлора и мы сравнили значения функций через различные методы суммирования и через библиотечные функции в math.h.

Рассмотрев два метода суммирования, я пришла к выводу, что они практически не отличаются друг от друга, различия незначительны. Для подсчета значений функций необходимо большее число слагаемых в сумме, метод суммирования не так важен.

# Литература

1. "Численные методы" — И. С. Березин, Н. П. Жидков.

2. "Язык программирования C" — Брайан Керниган, Деннис Ритчи.

# Приложение

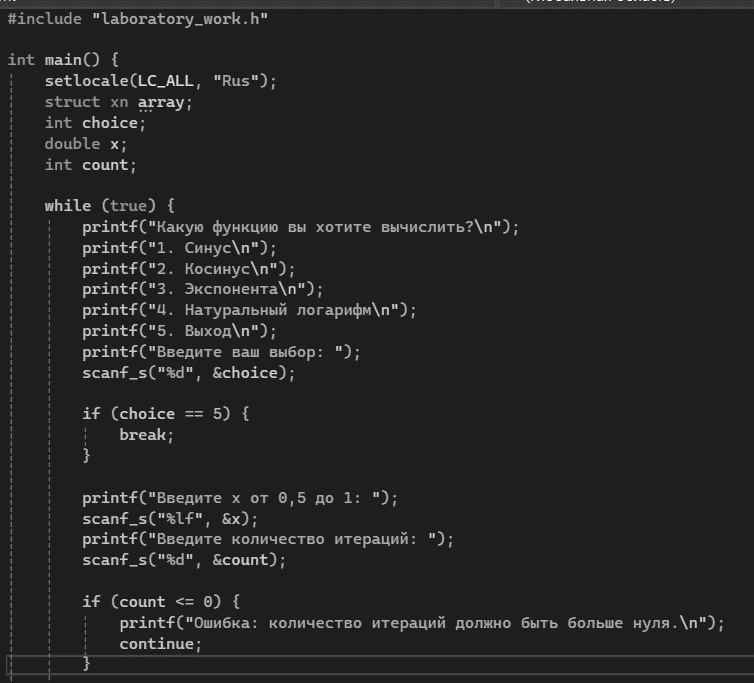


Рис. 16. Функция "main" часть 1

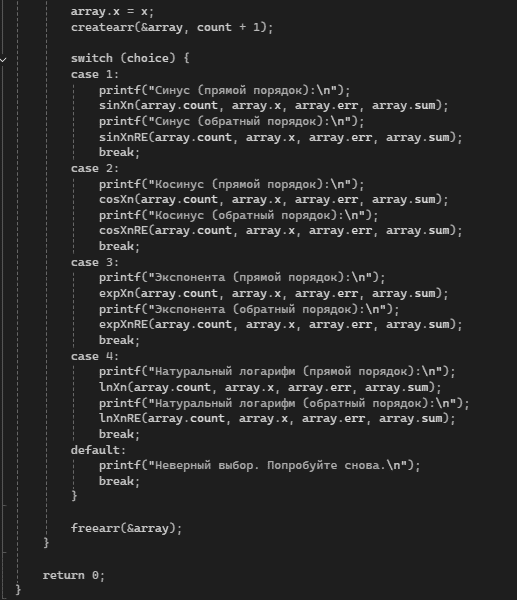


Рис. 17. Функция "main" часть 2