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**

**«Интерпретатор C-подобного языка программирования»**

**Выполнили**:

студенты группы 3823Б1ПМ1-1

Загрядсков М.А., Болтенков С.С.

**Проверил**:

преподаватель каф. ВВСП,

Волокитин В.Д.

Нижний Новгород

2025

**Содержание**

[Постановка задачи 3](#_Toc152527359)

[Метод решения 4](#_Toc152527360)

[Руководство пользователя 7](#_Toc152527361)

[Описание программной реализации 8](#_Toc152527362)

[Подтверждение корректности 9](#_Toc152527363)

[Результаты экспериментов 10](#_Toc152527364)

[Заключение 16](#_Toc152527365)

[Приложение 17](#_Toc152527366)

[Источники 18](#_Toc152527367)

# Постановка задачи

Требовалось:

1. Спроектировать и реализовать программу, способную проинтерпретировать (выполнить по отдельным командам) программу на C-подобном языке. Должно поддерживаться максимально возможное количество ключевых слов и операций из языка C98.
2. Протестировать программу на нескольких примерах с использованием всех поддерживаемых ключевых слов и операций. Убедиться в корректности обработки ошибок, которые могут быть в пользовательской программе.

# Руководство пользователя

Взаимодействие с исходной и пользовательской программами осуществляется через консоль. Пользовательская программа должна быть описана в файле code.cpp в папке проекта code. Вывод ошибок, а также ввод/вывод пользовательской программы осуществляется также через консоль.

Поддерживаемые команды:

1. Операции с переменными:
   1. Логические операции: ==, <=, >=, <, >, !=. Любая переменная приводима к логическому типу
   2. Арифметические операции: =, +, +=, -, -=, \*, \*=, /, /=, % (для целочисленных типов данных), %= (для целочисленных типов данных).
2. Типы данных: void (для возвращаемых значений функций), int, string, double.
3. Условные операторы: if, elif, else.
4. Операторы циклов: for, while. В for поддерживается объявление итерационной переменной в условии for.
5. Стандартные предопределенные функции:
   1. print(var1, var2, …, varn) – выводит в консоль n переменных через пробел, после выводит специальный символ новой строки.
   2. scan(var1, var2, …, varn) – запрашивает из консоли n переменных через клавишу новой строки (enter).
   3. Математические функции: abs(var), sin(var), cos(var), sqrt(var).
6. Прочие ключевые слова и специальные символы:

# Описание программной реализации

В папке проекта доступны следующие файлы:

1. Заголовочный файл с описанием функций, используемых при вычислении рядов sumseries.hpp.
2. Файл реализации этих функций sumseries.cpp.
3. Файл решения Visual Studio MathLabProg.sln.
4. Файл проекта Visual Studio MathLabProg.vcxproj.
5. Файл описания собственных математических функций \_math.hpp.
6. Файл реализации этих функций \_math.cpp.
7. Файл MathLabProg.cpp – код программы, позволяющей взаимодействовать с пользователем через консоль.
8. Файл MathLabExperiments.cpp – код программы, тестирующей функции на корректность и записывающей погрешности в файл.
9. Файл Отчет\_лабораторная\_МатФункции\_Загрядсков\_Максим\_3823Б1ПМ1-1.docx – данный файл с текстом отчёта.

# Подтверждение корректности

Для подтверждения корректности в программе значения функций сравнивались с значениями тех же функций при тех же аргументах, реализованных в библиотеке cmath. В результате при программа относительно точно вычисляла значения данных функций.

# Результаты экспериментов

Эксперименты проводились следующим образом: выбирался отдельный метод суммирования, затем происходило вычисление всех 4-х функций, после чего в файл записывалась разность значения функции из cmath и функции из \_math для одного аргумента. Вычисления производились для типа данных float, аргумент принимал значения от 0 до 64 с шагом в .

После первых нескольких экспериментов было обнаружено, что для слишком маленьких значений – количества итераций – точность сильно снижается, а при больших значениях функция обратного суммирования вычисляла ряд с большой ошибкой. Поэтому эмпирически была получена функция зависимости количества итераций от значения переменной , где – некоторые константы.

Сперва были получены точечные диаграммы разности значения функции из библиотеки cmath и значения функции суммирования для каждой функции и каждого типа суммирования на разных масштабах. Приведу некоторые из них:

*Рис. 1. Диаграммы разности значений прямой и попарной суммы для функции синуса и значений функции синуса из cmath.*

Как можно заметить здесь, попарная сумма синуса начинает расходиться немного раньше прямой суммы. Как можно будет увидеть позже, прямая сумма для синуса точнее, чем попарная.

*Рис. 2. Диаграммы разности значений обратной суммы функции синуса и обратной суммы функции синуса с массивом и значений функции синуса из cmath.*

На данных диаграммах видно, что обратная сумма синуса без массива начинает раньше расходиться и обладает большей погрешностью вблизи 0. Это может быть связано с тем, что вблизи нуля значения синуса малы, поэтому погрешность получения последнего члена и погрешность деления для получения старших членов в сумме наиболее заметны.

*Рис. 4. Диаграммы разности значений прямой суммы функции натурального логарифма и обратной суммы функции натурального логарифма с массивом и значений функции натурального логарифма из cmath.*

На данных диаграммах видно, что для обоих типов суммирования характерна большая погрешность при маленьких значениях аргумента. Но для обратной суммы погрешность гораздо больше и доходит до значений . Я связываю это с неверным получением последнего члена для суммы.

*Рис. 4. Диаграммы разности значений прямой суммы функции синуса, косинуса и натурального логарифма и значений функций синуса, косинуса и натурального логарифма из cmath.*

Как видно из диаграмм, все суммы принимают наиболее точные значения вблизи нулей функций.

Остальные диаграммы смотрите в приложении.

Следующим этапом в исследовании было сравнение точности суммирований. Для этого среди 4х значений разности выбиралось ближайшее к нулю, а тот способ, который достигал этого значений, получал один балл. Способ, набравший наибольшее количество баллов лучше вычислял значение функции. Приведу сюда таблицу сумм баллов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Функция | Прямая сумма | Попарная сумма | Обратная сумма | Обратная сумма с массивом |
| Синус | 26653 | 16386 | 25583 | 25583 |
| Косинус | 17789 | 14718 | 21753 | 17812 |
| Экспонента | 20678 | 25743 | 10335 | 38513 |
| Логарифм | 16061 | 24078 | 14952 | 45766 |

*Рис. 5. Таблица сумм баллов за наиболее точное вычисление различных функций для различных способов суммирования. Зеленым цветом обозначен наиболее точный для данный функции способ суммирования, голубым – второй по точности.*

По итогам исследования можно заметить, что для синуса наиболее точной оказалась прямая сумма, немного менее точными стали обе обратные суммы. Для косинуса же обратная сумма была заметно точнее прямой суммы и обратной суммы с массивом. Данные между синусом и косинусом так различаются, поскольку вблизи точки 0 косинус находится вблизи 1, что понижает общую точность функций. Попарная сумма оказалась наименее точной, так как для этих функций характерна разность знаков соседних членов, а попарная сумма эффективна только в случае, если соседние члены достаточно разных порядков. В случае с тригонометрическими функциями, сами попарные суммы оказывались разных порядков больше, чем обычные члены в прямой сумме. Прямая сумма оказалась формально точнее обратных для синуса за счёт значений при больших . Для косинуса прямая сумма и обратная сумма с массивом оказались практически равной точности, а так как обратная сумма имеет свойство занижать результат, то при больших она оказалась формально точнее.

Для экспоненты и логарифма ситуация другая: поскольку для обеих функций не характерна разница знаков в соседних членов, но характерна быстрая сходимость, попарная сумма оказалась довольно точной. Обратная сумма с массивом стала наиболее точной как при маленьких , за счет точного вычисления последних битов, так и при больших , за счет точных первых битов и порядка благодаря первому члену суммы, который не суммировался с соседним как в случае с попарной суммой.

# Заключение

По результатам проведенных экспериментов были сделаны следующие выводы:

* Все способы недостаточно точны из-за самого принципа многочлена Маклорена: чем дальше от , тем меньше точность. Из-за этого при достаточно больших некоторые способы оказывались формально точнее, хотя все они были далеки от действительного значения.
* Для расчёта тригонометрических функций довольно точным оказался способ прямого суммирования из-за разницы знаков соседних членов. Для расчёта экспоненты и логарифма точным оказался способ попарного суммирования из-за большой разности между соседними членами. Для всех 4-х функций точным был способ обратного суммирования с массивом. Обычный способ обратного суммирования был менее точным из-за погрешности получения последнего члена.

# Приложение