Решение систем линейных уравнений методом Крамера

Автор проекта: Ооржак Эрес Аясович

Группа: 01151 ДБ

Дата: 18.12.2024

Ведущий преподаватель: Усенко Олег Валерьевич

ВВЕДЕНИЕ

В данном проекте была разработана программа для решения систем алгебраических линейных уравнений методом Крамера. Программа предоставляет пользователю интерфейс для ввода системы уравнений и выводит решение вектора переменных. Решение осуществляется с помощью детерминантов и замены столбцов матрицы.

Основные особенности программы:

# Поддержка решений систем линейных уравнений любой размерности.

# Валидация ввода данных, чтобы избежать ошибок при вводе чисел.

# Алгоритм решения методом Крамера, который требует вычисления детерминантов для каждой переменной.

Типы данных

Для представления системы линейных уравнений используется структура «LinearSystem». Эта структура представлена на рисунке 1.

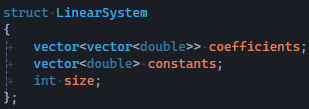


Рисунок 1 – Структура «LinearSystem»

1. Coefficients – матрица коэффициентов системы уравнений. Это двумерный вектор, где каждый элемент является коэффициентом уравнения.
2. Constants – вектор правой части системы. Это одномерный вектор, содержащий значения правой части каждого уравнения.
3. Size – размерность системы, т.е. количество уравнений и переменных.

Описание ключевых функций и алгоритмов

В проекте всего использовалось восемь функций. Первая из них это «CinStreamClear». Реализация функции представлена на рисунке 2.

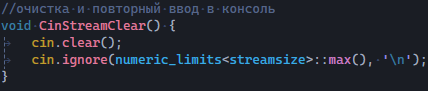


Рисунок 2 – Функция повторного ввода в консоль

Эта функция позволяет пользователю повторно вводить данные с клавиатуры, если он сделал до этого неверно.

Алгоритм очистки реализуется так:

1. Первый делом очищается поток ввода данных с клавиатуры.
2. После чего поток ввода игнорирует все введённые символы до новой строки.

Второй функцией в проекте стала функция ввода для пользователя. А именно «UserInput». Реализация функции представлена на рисунке 3.



Рисунок 3 – Реализация функции ввода

Эта функция позволяет пользователю вводить размерность системы, коэффициенты матрицы и вектор правой части.

Алгоритм ввода реализуется так:

1. В функцию аргументом передаётся ранее созданная структура, чтобы мы могли работать конкретно с элементами структуры. Структура передаётся с помощью адреса, чтобы не делать копию.
2. В теле функции создаётся переменная типа «int» для хранения количества попыток пользователя. В теле цикла «while» стоит условие, пока попытки будут меньше 3 будет выполняться тело цикла. Изначально пользователь должен ввести размерность системы, и вовремя ввода размерности системы идёт проверка на корректность ввода. Если пользователь ввёл букву, отрицательное число или 0, то ему выводится ошибка и увеличиваются попытки.
3. После ввода размерности идёт ввод самой матрицы. Алгоритм ввода матрицы реализован в два цикла «for». У первого цикла счётчик занимается строками, у второго счётчик занимается столбцами. После чего идёт ввод в двумерный вектор структуры. И если была ошибка ввода, то пользователя уведомляют об этом.
4. Как только пользователь закончил ввод матрицы, сразу же после этого идёт ввод правого вектора. Так как правый вектор – это одномерный вектор, для ввода чисел требуется только один цикл «for». Все остальное в алгоритмах идентично.

Третьей функцией в проекте стала функция для подсчёта детерминанта матрицы. А конкретно «ComputeDeterminant». Реализацию функции можно видеть на рисунке 4.



Рисунок 4 – Первая реализация функции детерминанта

В самом начале функции я создал переменную типа «int» для хранения размера матрицы. После чего уже пошла реализация подсчёта детерминанта. В первых простейших случаях, когда размер матрицы не больше, чем 3 на 3, идёт автоматический подсчёт, без участия других алгоритмов. Как только размерность матрицы достигает или превышает 4 идёт алгоритм Гаусса, для приведения матрицы к верхнетреугольному виду. Алгоритмическая сложность, а именно «O большое» этого алгоритма составляет x3.

Но есть и другое решение, а именно алгоритм миноров. Реализация этого алгоритма представлена на рисунке 5.



Рисунок 5 – Вторая реализация функции детерминанта

Алгоритм работает так, что начинает с первой строчки, а именно с 0. Алгоритм не учитывает числа максимально близкие к нулю, с погрешность 10-9. После чего создаётся подматрица, которая создаётся вычёркиванием строки столбца. И уже после идёт рекурсия вызова самой функции в теле этой же функции.

Четвёртой функцией является функция «ReplaceColumn». Реализацию функции можно посмотреть на рисунке 6.

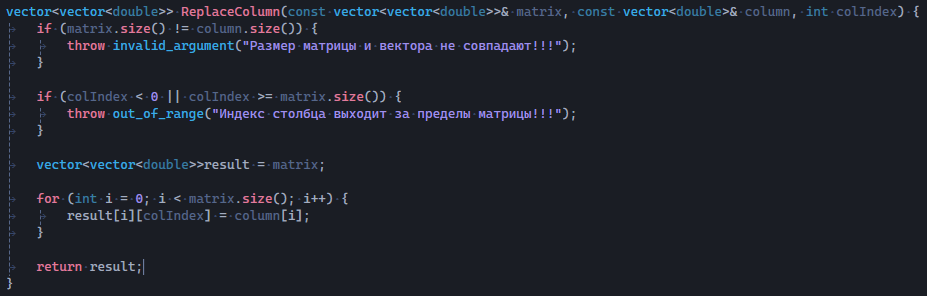


Рисунок 6 – Реализация функции замены

Функция принимает оригинальную матрицу, вектор-столбец, который будет заменять столбец в матрице и индекс столбца. Первым делом идёт проверка того, что совпадает ли размер столбца матрицы и вектор. После чего идёт проверка индекса столбца. Затем создаётся матрица, в которую копируется оригинальная матрицы. И в самом конце идёт замена указанного вектора-столбца матрицы на вектор-столбец. И возвращается полученная матрицы.

Пятой функцией является функция «SolveSystem», которая уже занимается решением систем линейных алгебраических уравнений методом Крамера. Реализацию функции можно увидеть на рисунке 7.

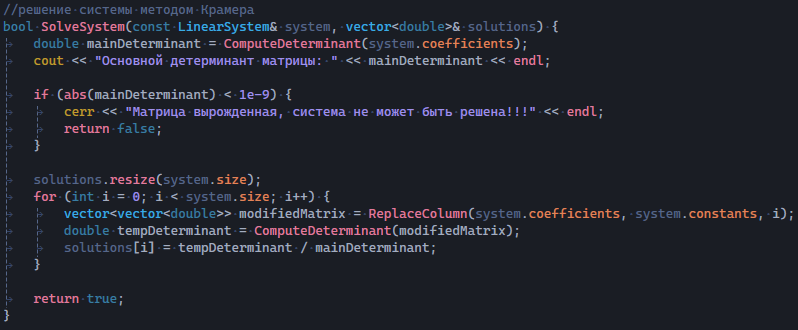


Рисунок 7 – Реализация функции Крамера

Функция в качестве аргументов принимает структуру линейной системы и вектор решений. Первым делом идёт подсчёт главного детерминанта матрицы после чего он выводится на экран. После чего идёт проверка вырожденности матрицы, с учётом машинной точности, т.е. точность 10-9. Если проверка прошла, то идёт подсчёт методом Крамера. Размер вектора решений подгоняется под размер матрицы. Для каждого неизвестного решения системы создаётся новая матрица с помощью функции «ReplaceColumn». Потом вычисляется детерминант для вновь созданных матрицы. И в конце уже высчитывается сами неизвестные методом Крамера.

Шестая функция выводит матрицу в красивом виде на консоль. Реализацию функции можно увидеть на рисунке 8.



Рисунок 8 – Реализация вывода матрицы

В качестве аргумента принимает вектор для матрицы. Первым делом создаётся переменная для хранения размера матрицы. После чего уже идёт проверка является ли матрицы пустой и вывод ответа на консоль. Потом циклом создаём рамку для матрицы и уже во втором цикле выводим элементы матрицы на консоль разделяя их символом «|». После чего создаётся нижняя рамка.

Седьмая функция выводит вектор на консоль. Реализацию можно увидеть на рисунке 9.

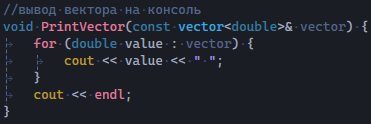


Рисунок 9 – Реализация вывода вектора

Восьмая функция узнаёт у пользователя, хочет ли он продолжить подсчёт неизвестных матрицы. Реализацию функции можно увидеть на рисунке 9.

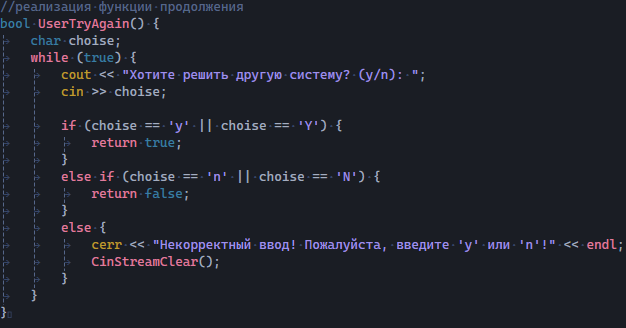


Рисунок 9 – Реализация функции продолжения работы

После того, как я показал и расписал все функции, которые были созданы в проекте, пришло время рассказать про функцию «main». Реализацию можно увидеть на рисунке 10.

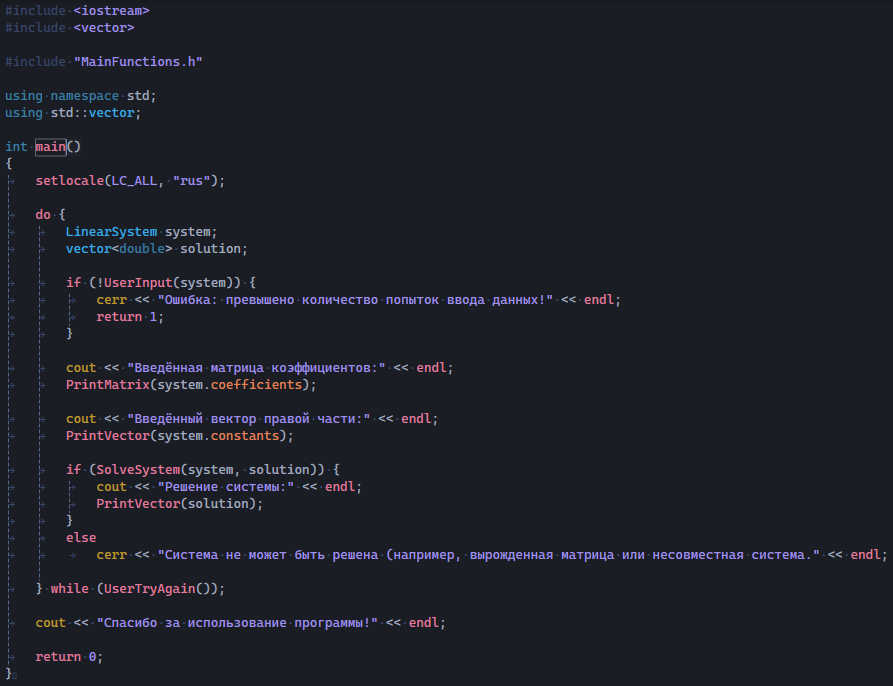


Рисунок 10 – Реализация main

В цикле «do-while» создаётся экземпляр структуры «LinearSystem» и вектор решений. Если пользователь ничего не ввёл, то ему выдаётся ошибка и ответ на экране. После чего на консоль выводится созданная матрица и вектор. И если система имеет решение, оно выводится на экран. Иначе на консоль выводится ошибка и ответ. В цикле «do-while» в теле «while» есть условие, что пока пользователь хочет снова решать системы. Когда пользователь нажмёт на «n» или «N», то выводится ответ спасибо.