Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

Операции над матрицами и решение типовых заданий аналитической геометрии

Студент гр. 953501

А.П. Харкевич

Руководитель ассистент кафедры информатики

И.А. Удовин

Минск 2019

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой Информатики

––––––––––––––––––––––––

(подпись)

Волорова Н.А. 2020 г.

ЗАДАНИЕ

по курсовому проекту

Студенту   *Харкевичу Антону Павловичу*  –––––––– –––––\_\_\_–

1. Тема работы *Операции над матрицами и решение типовых заданий аналитической геометрии* \_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-\_––––\_\_

2. Срок сдачи студентом законченной работы 3*31.05.2020 г*.–––   \_\_\_\_ \_\_\_\_\_-

3. Исходные данные к работе *Операционная система Windows. Язык программирования C#.*

4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение. 1. Анализ предметной области. 2. Разработка программного средства. 3. Тестирование и проверка работоспособности. Заключение. Список использованных источников..*

5. Консультант по курсовой работе *Удовин И. А.*

6. Дата выдачи задания *01.02.2020 г.*

7. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

*раздел 1, Введение к 28.02.2020г. – 10 % готовности работы;*

*раздел 2 к 15.03.2020г. – 30 % готовности работы;*

*раздел 3 к 15.04.2020г. – 60 % готовности работы;*

*раздел 4 к 10.05.2020г. – 80 % готовности работы;*

*Заключение, Приложения к 20.05.2020г.– 90 % готовности работы; оформление пояснительной записки и графического материала к 24.05.2020г. – 100 % готовности работы.*

*Защита курсового проекта с 31.05.2020 г. по 01.06.2020 г. .*

РУКОВОДИТЕЛЬ *Удовин И. А.*

(подпись)

Задание принял к исполнению *Харкевич А.П. 01.02.2019 г.* (дата и подпись студента)

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 5](#_Toc10197997)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 8](#_Toc10197998)

[1.1 Особенности работы с матрицами 8](#_Toc10197999)

[1.2 Особенности заданий аналитической геометрии 9](#_Toc10198000)

[1.3 Выбор инструментария 9](#_Toc10198001)

[1.4 Постановка задачи 10](#_Toc10198002)

[2. РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА 11](#_Toc10198003)

[2.1 Сущность интерфейса 11](#_Toc10198004)

[2.2 Реализация работы с матрицами 11](#_Toc10198005)

[2.3 Реализация работы с векторами 12](#_Toc10198006)

[2.4 Реализация нахождения базовых элементов треугольника 14](#_Toc10198007)

[3. ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ 15](#_Toc10198009)

[3.1 Тест 1 15](#_Toc10198010)

[3.2 Тест 2 16](#_Toc10198011)

[3.3 Тест 3 16](#_Toc10198012)

[3.4. Тест 4 17](#_Toc10198013)

[3.5. Тест 5 17](#_Toc10198014)

[3.6. Тест 6 18](#_Toc10198014)

[3.7. Тест 7 18](#_Toc10198014)

[3.8. Тест 8 19](#_Toc10198014)

[3.9. Тест 9 19](#_Toc10198014)

[ЗАКЛЮЧЕНИЕ 20](#_Toc10198015)

[СПИСОК ИСТОЧНИКОВ 20](#_Toc10198015)

# 

# **ВВЕДЕНИЕ**

Такой раздел высшей математики, как линейная алгебра, появился ещё очень давно.

Первые элементы линейной алгебры следовали из практических вычислительных задач вокруг решения линейных уравнений. Близкие к современным матричным методам подходы к решению систем линейных уравнений обнаруживаются у вавилонян.

Методы, сформировавшие линейную алгебру как самостоятельную отрасль математики, уходят корнями в другие разделы. Ферма в 1630-е годы, создав классификацию плоских кривых, ввёл в математику (ключевой для линейной алгебры) принцип размерности и разделил задачи аналитической геометрии по числу неизвестных Эйлер создал классификацию кривых по порядкам обратив внимание на линейный характер преобразований координат, ввёл в оборот понятие аффинного преобразования.

Первое введение понятия определителя для целей решения систем линейных уравнений относят к Лейбницу, но эти работы не были опубликованы. Крамер и Безу в работах по проблеме отыскания плоской кривой, проходящей через заданную точку, вновь построили это понятие (правило Крамера сформулировано в 1750 году), Вандермонд и Лагранж дали индуктивное определение для случаев, а целостное определение и окончательные свойства определителей дали Коши (1815) и Якоби (1840-е годы). Гауссу (около 1800 года) принадлежит формализация метода последовательного исключения переменных для решения этих задач, ставшего известным под его именем.

Ещё одним источником подходов для линейной алгебры стала проективная геометрия. Но основной базой линейной алгебры стало фактически влившееся в раздел векторное исчисление, очерченное Гауссом в работах по геометрической интерпретации комплексных чисел (1831) и обретшее окончательную форму в трудах Мёбиуса, Грассмана и Гамильтона 1840-х — 1850-х годах. Физики школы Гамильтона, из которых самым выдающимся был Максвелл, тщательно проработали то, что сейчас относится к векторной алгебре в трёхмерном евклидовом пространстве: введены понятия скалярного, векторного и смешанного произведений векторов, набла-оператор, сформирована вошедшая в традицию символика, также начиная с этого времени векторы проникают и в школьные программы.

Параллельно шло развитие линейной алгебры и в Европе. В 1844 году Грассман строит понятие внешней алгебры, описывающей подпространства линейного пространства. Долгое время его работы незаслуженно обходились вниманием: языком, адекватным физической картине мира, считался язык кватернионов. Синтез идей Грассмана и Гамильтона был осуществлён в 1870-х Клиффордом: введённое им понятие алгебры Клиффорда включает как частные случаи как алгебру кватернионов, так и внешнюю алгебру.

Понятие матрицы ввёл Сильвестр в 1850 году. Кэли обстоятельно разрабатывает матричное исчисление, публикуя в 1858 году «Мемуар о теории матриц». Системы линейных уравнений в матрично-векторном виде впервые появились, по-видимому, в работах Лагерра (1867). Матричные группы, связанные с неевклидовыми геометриями, появились в работах Киллинга в 1880-х годах, вместе с более ранними работами Ли они стали основой теории групп и алгебр Ли. На рубеже веков эта теория была обогащена Энгелем и Картаном, давшими классификацию полупростых алгебр Ли и попутно открывшими векторное произведение в семимерном пространстве.

Теория инвариантов в классическом варианте — учение о свойствах алгебраических форм, сохраняющихся при линейных преобразованиях, сформирована начиная с 1840-х годов в работах Кэли, Эрмита и Сильвестра (известных как «инвариантная троица», фр. la trinité invariantive), считается, что именно теория инвариантов и приводит к созданию принципов решения произвольных систем линейных уравнений. В частности, Эрмит[уточнить] сформулировал и решил в частном случае проблему нахождения системы линейных диофантовых уравнений, решение в общем случае найдено Смитом (англ. Henry John Stephen Smith), результат которого остался незамеченным, пока не был обнаружен в 1878 году Фробениусом. Финальный вид результаты о системах линейных уравнений с произвольными числовыми коэффициентами получили в работах, организованных Кронекером, в которых принимали участие Вейерштрасс, Фробениус и группа немецких учёных, особое внимание уделялось строгости и точности формулировок. В частности, определитель в курсе лекций Кронекера — Вейршртаса вводился как полилинейная знакопеременная функция от n векторов n-мерного пространства, нормированная таким образом, что принимает значение 1 для единичной матрицы; притом это определение эквивалентно вытекающему из исчисления Грассмана. Фробениус в 1877 году ввёл понятие ранга матрицы, основываясь на котором в ближайшие годы сразу несколько учёных доказали утверждение об эквивалентности разрешимости системы линейных уравнений совпадением рангов её основной и расширенной матрицы, известной в русских и польских источниках как теорема Кронекера — Капелли, во французских — теорема Руше (фр. Eugène Rouché) — Фонтене (фр. Georges Fontené), в немецких и испанских — теорема Руше — Фробениуса, в итальянских и английских — теорема Руше — Капелли.

В 1888 году Пеано на базе исчисления Грассмана впервые в явном виде сформулировал аксиомы линейного пространства (векторных пространств над полем действительных чисел в том числе бесконечномерных) и применил обозначения, сохранившиеся в употреблении в XX—XXI века. Тёплиц в начале 1910-х годов обнаружил, что при помощи аксиоматизации линейного пространства для доказательства основных теорем линейной алгебры не требуется прибегать к понятию определителя, что позволяет распространить их результаты на случай бесконечного числа измерений, иными словами, линейная алгебра применима при любом основном поле. Аксиоматическое определение векторного и евклидова пространства было впервые чётко сформулировано в начале XX века практически одновременно Вейлем и фон Нейманом, исходя из запросов квантовой механики.

Со второй половины XX века с появлением компьютеров, развитием методов вычислительной математики и компьютерной алгебры в рамках линейной алгебры получило бурное развитие вычислительное направление — отыскание методов и алгоритмов, обеспечивающих эффективное решение задач линейной алгебры с использованием вычислительной техники, сформировался самостоятельный раздел вычислительной линейной алгебры.

**1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1 Ключевые понятия и особенности работы с матрицами**

Матрица — математический объект, записываемый в прямоугольной таблице размером MxN, в ячейках которой расположены элементы произвольного заранее выбранного (основного) поля (в наиболее общем случае — ассоциативного кольца — это могут быть целые, вещественные или комплексные числа, векторы, рациональные функции — в зависимости от приложений и задач.

Матрицы широко применяются в математике для компактной записи систем линейных алгебраических или дифференциальных уравнений. В этом случае количество строк матрицы соответствует числу уравнений, а количество столбцов — количеству неизвестных. В результате решение систем линейных уравнений сводится к операциям над матрицами. Для матрицы определены следующие алгебраические операции:

сложение матриц, имеющих один и тот же разме;

умножение матриц подходящего размера (матрицу, имеющую n столбцов, можно умножить справа на матрицу, имеющую n строк.

в том числе умножение на матрицу вектора (по обычному правилу матричного умножения; вектор является в этом смысле частным случаем матрицы;

умножение матрицы на элемент основного кольца или поля (то есть скаляр).

Относительно сложения матрицы образуют абелеву группу; если же рассматривать ещё и умножение на скаляр, то матрицы образуют модуль над соответствующим кольцом (векторное пространство над полем). Множество квадратных матриц замкнуто относительно матричного умножения, поэтому квадратные матрицы одного размера образуют ассоциативное кольцо с единицей относительно матричного сложения и матричного умножения.

Доказано, что каждому линейному оператору, действующему в n-мерном линейном пространстве, можно сопоставить единственную квадратную матрицу порядка n; и обратно — каждой квадратной матрице порядка n может быть сопоставлен единственный линейный оператор, действующий в этом пространстве. Свойства матрицы соответствуют свойствам линейного оператора.

Определитель — многочлен, комбинирующий элементы прямоугольной матрицы особым способом, благодаря которому независимо от транспонирования и линейных комбинаций строк или столбцов характеризуется содержание матрицы; в частности, если в матрице есть линейно-зависимые строки или столбцы — определитель равен нулю. Квадратные матрицы, определитель которых равен нулю называются вырожденными, для них не определено обращение; если определитель отличен от нуля — то матрица называется невырожденной. Определитель играет ключевую роль в решении систем линейных уравнений в общем виде, на его базе вводятся понятия минора, дополнительного минора, алгебраического дополнения.

Так как мы заранее не знаем размерность матрицы, чаще всего в работе с ними будут использоваться итерационные алгоритмы. Такие операция, как нахождение определителя, будет выполняться рекурсивно.

## **1.2 Основные понятия и особенности заданий аналитической геометрии**

Понятие вектора (сам термин «вектор» был введён У. Гамильтоном) изначально возникло как геометрическая абстракция для объектов, характеризующихся одновременно величиной и направлением, таких как скорость, момент силы, напряжённость электрического поля, намагниченность. В начале XX века изначальная интерпретация векторов (до сих пор используемая в элементарной математике) как «направленных отрезков» сменилось на аксиоматику векторного пространства с двумя операциямиː сложением векторов и умножение вектора на числа (более общо, на элементы поля). Кроме того, часто вводятся различные виды произведения векторов: скалярное, векторное, смешанное.

Задания аналитической геометрии требуют высокой вычислительной точности. Так как ключевое слово в аналитической геометрии – аналитическая (т.е. для решения заданий нужно сначала проанализировать ситуацию), то не для всех задач можно составить алгоритм программного вычисления. Для создания алгоритмов к некоторым задачам придётся проявить творческий подход.

**1.3 Выбор инструментария**

Для реализации курсовой работы была выбрана интегрированная среда разработки Microsoft Visual Studio 2019, для разработки на языке С#.

Microsoft Visual Studio 2019 позволяет разрабатывать приложения с графической оболочкой с поддержкой технологии Windows Forms, которая будет использоваться в данной курсовой работе, потому что позволяет быстро и легко создать удобный интерфейс. Так же в данной среде разработки есть удобный Just-In-Time отладчик кода, позволяющий легко обнаружить ошибки в коде.

Язык С# – объектно-ориентированный язык программирования и является языком разработки приложений для платформы Microsoft .NET Framework и Microsoft .NET Core. Язык был выбран потому что он является одним из языком, изучаемых во втором семестре.

Git – распределённая система контроля версий, для запасного плана, в случае надобности отката проекта на предыдущую версию.

GitHub — веб-сервис для хостинга IT-проектов и их совместной разработки. Для хранения курсовой работы не только на компьютере.

## **1.4 Постановка задачи**

В задачу курсовой работы входит разработать программное средство на языке C#, которое:

* Сможет выполнять основные операции над матрицами
* Сможет решать типовые задания аналитической геометрии, такие как нахождение всех элементов треугольника по координатам трёх его точек.
* Сможет выполнять основные операции над векторами
* Сможет составлять уравнения основных элементов линейного пространства.
* Сделать это программное средство не затратным по ресурсам, чтобы его могли использовать даже слабые компьютеры.
* Разработать простой и удобный пользователю интерфейс программного средства.

Другими словами, разработать математический калькулятор с надлежащим ему функционалом.

**2. РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

**2.1 Сущность интерфейса**

Для создания легкого в пользовании интерфейса приложения были использованы несколько критериев:

1. Интерфейс не только должен помочь найти нужную информацию или быстро разобраться с функциями, но и доставить пользователю эстетическое удовольствие.
2. Не добавлять ничего лишнего. При создании интерфейса приложения это правило действует, правда, с поправками. Нужно сокращать количество элементов до минимума пока это возможно, но не в ущерб задачам, которые решает пользователь.
3. Все взаимосвязанные элементы логически соединены.

Пользовательский интерфейс данного программного средства представляет собой консольное приложение. Оно реализовано в виде меню со множеством подпунктов. Из каждого подпункта возможен выход в главное меню. Данное меню реализовано с помощью множественных вложенных конструкций switch. Интерфейс прост и понятен пользователю. В каждом из подпунктов есть подсказки по тому, какие клавиши нажимать. Везде, где это возможно, идёт проверка корректности ввода, чтобы программа не теряла работоспособности. Есть проверка исключительных ситуаций (например, деление на ноль).

## **2.2 Реализация работы с матрицами**

Работа с матрицами реализуется через статический класс WorkWithMatrix в котором хранятся следующие методы:

* MatrixOutput – вывод матрицы на консоль
* MatrixAddition – сложение двух матриц
* MatrixSubstraction – вычитание двух матриц
* MatrixMultiplication – умножение двух матриц
* MatrixMultiplicationByNumber – умножение матрицы на заданное число
* MatrixDivisionByNumber – деление матрицы на заданное число
* MatrixTr – вычисление следа матрицы (суммы элементов, стоящих на главной диагонали квадратной матрицы)
* MatrixTranspose – транспонирование матрицы
* Determinant – вычисление определителя квадратной матрицы
* InverseMatrix – вычисление обратной матрицы для заданной матрицы
* TransitionToNewCoordinates – преобразование заданной матрицы к матрице в новом базисе с помощью матрицы перехода к новым координатам
* SolvingSystemsOfLinearEquations – решение систем линейных уравнений методом крамера

**2.3 Реализация работы с векторами**

Работа с матрицами реализуется через статический класс WorkWithVectors в котором хранятся следующие методы:

* VectorLength – вычисляет длину заданного вектора
* VectorOrt – вычисляет орт (единичный вектор) для заданного вектора
* VectorMultiplicationNumber – производит умножение вектора на заданное число
* VectorDivisionNumber – производит деление вектора на заданное число
* VectorAddition – производит суммирование двух заданных векторов
* VectorSubstraction – вычисляет разность для двух заданных векторов
* VectorScalarMultplication – вычисляет скалярное произведение для двух заданных векторов
* VectorGuideCosines – находит направляющие косинусы для заданного вектора
* VectorGuideCoordinates – вычисляет координаты вектора по двум заданным точкам
* SegmentLength – вычисляет длину отрезка между двумя заданными точками
* DivisionOfSegment – находит точку, которая делит заданный отрезок в заданном отношении
* AngleBetweenTwoVectors – находит угол между двумя заданными векторами
* CosOfAngleBetweenTwoVectors – находит косинус угла между двумя заданными векторами
* ProectionOfVectorOnOtherVector – находит проекцию заданного вектора на другой заданный вектор
* IsLinesParallel – позволяет узнать, параллельны ли две заданные прямые
* IsPlanesParallel – позволяет узнать, параллельны ли две заданные плоскости
* IsLinesPerpendicular – позволяет узнать, перпендикулярны ли две заданные прямые
* IsPlanesPerpendicular – позволяет узнать, перпендикулярны ли две заданные плоскости
* IsLineAndPlaneParallel – позволяет узнать, параллельны ли заданные прямая и плоскость
* IsLineAndPlanePerpendicular – позволяет узнать, перпендикулярны ли заданные прямая и плоскость
* AngleBetweenTwoPlanes – находит угол между двумя заданными плоскостями
* AngleBetweenTwoLines – находит угол между двумя заданными прямыми
* AngleBetweenLineAndPlane – находит угол между заданными прямой и плоскостью
* VectorOrtogonalToTwo – находит вектор, ортогональный двум заданным
* VectorOrtogonalToTwo – находит вектор, ортогональный двум заданным
* TwoPointLineEquation – строит уравнение прямой по двум заданным точкам
* PointAndVectorPlaneEquation – строит уравнение плоскости по заданному нормальному вектору и точке
* TwoVectorsAndPointPlaneEquation – строит уравнение плоскости по заданному вектору, параллельному двум и другим, и заданной точке
* ThreePointsPlaneEquation – строит уравнение плоскости по трём заданным точкам
* DistanceFromPointToPlane – находит расстояние от заданной точки до плоскости.
* ProectionOfPointOnPlane – находит проекцию точки на заданную плоскость
* DistanceBetweenCrossingLines – находит расстояние между скрещивающимися прямыми.
* VectorMultiplicationOfTwoVectors – находит векторное произведение двух заданных векторов
* MixedMultiplicationOfTwoVectors – находит смешанное произведение трёх заданных векторов
* GramSchmidtOrthogonalizationProcess – с помощью процесса ортогонализации Грамма-Шмидта строит ортонормированный базис из трёх векторов

**2.4 Реализация нахождения базовых элементов треугольника**

Нахождение базовых элементов треугольника было вынесено в отдельный класс WorkWithMatrix в котором хранятся следующие свойства:

* CoordinatesAB – возвращает координаты стороны стороны AB треугольника ABC
* CoordinatesAC – возвращает координаты стороны стороны AC треугольника ABC
* CoordinatesBC – возвращает координаты стороны стороны BC треугольника ABC
* LengthAB – возвращает длину стороны стороны AB треугольника ABC
* LengthAС – возвращает длину стороны стороны AС треугольника ABC
* LengthBС – возвращает длину стороны стороны BС треугольника ABC
* AngleA – возвращает значение в градусах угла A треугольника ABC
* AngleB – возвращает значение в градусах угла B треугольника ABC
* AngleC – возвращает значение в градусах угла C треугольника ABC
* Perimetr– возвращает величину периметра треугольника ABC
* Area– возвращает величину площади треугольника ABC
* RasiudOfSmallCircle– возвращает длину вписанной в треугольник ABC окружности
* RasiudOfSmallCircle– возвращает длину описанной около треугольника ABC окружности

**3. ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ**

В ходе тестирования были проверены ключевые функции, из которых следует работоспособность всего программного средства.

**3.1 Тест 1**

|  |  |
| --- | --- |
| Тестовая ситуация | Использование любой функции для работы с матрицей (например, сложение). |
| Ожидаемый результат | Две матрицы сложатся и будет выведена их сумма. |
| Фактический результат | Все работает корректно. |

**3.2 Тест 2**

|  |  |
| --- | --- |
| Тестовая ситуация | Использование любой функции для работы с векторами (например, скалярное произведение двух векторов). |
| Ожидаемый результат | Будет выведено скалярное произведение двух векторов. |
| Фактический результат | Все работает корректно. Демонстрация: |

**3.3 Тест 3**

|  |  |
| --- | --- |
| Тестовая ситуация | Нахождение какого-нибудь из элементов треугольника (например, площади) при задании его по трём точкам. |
| Ожидаемый результат | Будет выведено корректное значение площади. |
| Фактический результат | Все работает корректо. Демонстрация: |

**3.4. Тест 4**

|  |  |
| --- | --- |
| Тестовая ситуация | Нестандартный ввод при работе с векторами (например, разделить вектор на ноль). |
| Ожидаемый результат | Программа выдаст предупреждение о том, что мы ввели неправильное значение для деления и продолжит работу. |
| Фактический результат | Все работает корректно. Демонстрация: |

**3.5. Тест 5**

|  |  |
| --- | --- |
| Тестовая ситуация | Нестандартный ввод при работе с матрицами (например, найти обратную матрицу для матрицы с вырожденным определителем). |
| Ожидаемый результат | Программа выдаст предупреждение о том, что такой матрицы не существует и продолжит работу. |
| Фактический результат | Все работает корректно. Демонстрация: |

**3.6. Тест 6**

|  |  |
| --- | --- |
| Тестовая ситуация | Нестандартный ввод при работе с матрицами (например, найти элемент треугольника, стороны которого не удовлетворяют неравенству Минковского). |
| Ожидаемый результат | Программа выдаст предупреждение о том, что такого треугольника не существует и продолжит работу. |
| Фактический результат | Все работает корректно. Демонстрация: |

**3.7. Тест 7**

|  |  |
| --- | --- |
| Тестовая ситуация | Некорректный ввод (например, ввести буквы вместо цифр). |
| Ожидаемый результат | Программа выдаст предупреждение о том, что был совершен некорректный ввод и продолжит работу. |
| Фактический результат | Все работает корректно. Демонстрация: |

**3.8. Тест 8**

|  |  |
| --- | --- |
| Тестовая ситуация | Некорректный ввод числа в оператор switch (например, ввести число, которое не отвечает ни за одну функцию). |
| Ожидаемый результат | Программа выдаст предупреждение о том, что был совершен некорректный ввод и продолжит работу. |
| Фактический результат | Все работает корректно. Демонстрация: |

**3.9. Тест 9**

|  |  |
| --- | --- |
| Тестовая ситуация | Некорректный ввод (например, ввести матрицу с большим количеством пробелов и не в порядке следования строчек и столбцов). |
| Ожидаемый результат | Программа всё равно считает матрицу и продолжит работу |
| Фактический результат | Все работает корректно. Демонстрация: |

Из данного тестирования можно сказать, что данное программное средство справляется с поставленными ему задачами.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы был разработан корректно работающий математический калькулятор, реализующий около 60 алгоритмов. В ходе разработки данного программного средства были получены знания по программированию классов в среде разработки Microsoft Visual Studio 2019 с помощью языка C#, а также были получены теоретические знания в сфере программирования на платформе .NET.

Также стоит отметить довольно быструю работу самого программного средства.

Как вывод можно сказать, что с поставленными задачами программное средство справляется и способно работать даже на старых компьютерах, главным критерием остается только наличие .NET Framework и операционной системы Windows на этом компьютере.

**СПИСОК ИСТОЧНИКОВ**

1. Высшая математика, Часть 1, Жевняк Р.М., Карпук А.А., 1984
2. Wikipedia [Электронный ресурс]: https://ru.wikipedia.org
3. Microsoft .NET Documentation [Электронный ресурс]: https://docs.microsoft.com/en-us/dotnet