

Министерство науки и высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по курсу «Программирование»
на тему «Инженерный калькулятор»

Выполнил:
студент группы 23ВВВ2:
Пырков Д. А.

Принял:
Голотенков Н.О.
04.06.24.

Пенза 2024

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ



М.А. Митрохин

« » 20 г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Морозову Денису Александровичу Группа 238882

Тема проекта: *Разработка программы сложной структуры методом нисходящего программирования.*

Исходные данные (технические требования) на проектирование

Разработать программу - приложение "интерный калькулятор". Программа должна выполнять базовые арифметические операции, возведение в степень, извлечение квадратного корня, вычисление основных тригонометрических функций.

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Схема данных, схема ресурсов системы, схема работы системы, иерархическая структура программы, схема взаимодействия программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания “ 5 ” феврале 2024

Дата защиты проекта “ 04 ” 06. 24.

Руководитель  / Голотыянов Н. О. /

Задание получил “ 5 ” 1 февраля 2024 г.

Студент Мирков Мирков. О. А.

Содержание

Введение	5
1. Постановка задачи	5
2. Выбор решения	6
3. Отладка и тестирование	8
4. Описание программы	11
5. Руководство пользователя	23
5.1 Интерфейс калькулятора	23
5.2 Подробное описание функций	25
Заключение	29
Список используемых источников	30
Приложение А	31
Приложение В	41

Введение

В современном мире научно-технического прогресса инженерные расчеты занимают центральное место в развитии технологий и инноваций. Инженеры разных специальностей ежедневно сталкиваются с задачами, требующими быстрого и точного вычисления сложных математических выражений. Инженерный калькулятор, как инструмент, существенно облегчает этот процесс, позволяя выполнять широкий спектр вычислений: от базовых арифметических операций до более сложных функций.

Цель данной курсовой работы — разработка и реализация инженерного калькулятора, способного решать различные задачи, возникающие в повседневной практике инженера. В рамках работы будет рассмотрена архитектура приложения, выбраны оптимальные алгоритмы для выполнения вычислений, а также реализован удобный пользовательский интерфейс. Особое внимание уделено вопросам точности вычислений, производительности и удобству использования.

Данная работа представляет интерес не только для студентов и специалистов в области инженерии, но и для всех, кто заинтересован в эффективных инструментах для выполнения сложных вычислений. Реализация проекта позволит улучшить навыки программирования и углубить понимание принципов разработки вычислительных приложений, а также даст возможность оценить потенциал современных технологий в создании эффективных и надежных инженерных инструментов.

1. Постановка задачи

Необходимо разработать приложение, представляющее собой инженерный калькулятор. Программа должна иметь интуитивно понятный интерфейс для пользователя и правильно выполнять все входящие в неё операции. Обязательными условиями являются:

- использование сложных типов данных – структур, массивов, списков, файлов.

способов создания классических приложений с помощью визуального конструктора в Visual Studio. Такие функции, как размещение визуальных элементов управления путем перетаскивания, упрощают создание классических приложений.

В Windows Forms можно разрабатывать графически сложные приложения, которые просто развертывать, обновлять, и с которыми удобно работать как в автономном режиме, так и в сети. Приложения Windows Forms могут получать доступ к локальному оборудованию и файловой системе компьютера, на котором работает приложение.

Windows Forms — это технология пользовательского интерфейса для .NET, представляющая собой набор управляемых библиотек, которые упрощают выполнение стандартных задач, таких как чтение из файловой системы и запись в нее. С помощью среды разработки, такой как Visual Studio, можно создавать интеллектуальные клиентские приложения Windows Forms, которые отображают информацию, запрашивают ввод пользователя и взаимодействуют с удаленными компьютерами по сети.

В Windows Forms форма — это визуальная поверхность, на которой выводится информация для пользователя. Обычно приложение Windows Forms строится путем добавления элементов управления в формы и создания кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия клавиш. Элемент управления — это отдельный элемент пользовательского интерфейса, предназначенный для отображения или ввода данных. В Windows Forms предусмотрено множество элементов управления, которые можно добавлять в формы. Например, элементы управления могут отображать текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы.

При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается событие. Приложение реагирует на эти события с кодом и обрабатывает события при их возникновении.

3. Отладка и тестирование

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2022. Программа обладает всеми средствами необходимыми при разработке и отладке программы. Для отладки использовались несколько возможностей VisualStudio: точка останова, трассировка, анализ содержимого переменных. Рассмотрим каждый из этих инструментов и способы их использования.

Точки останова (Breakpoints)

Точка останова — это инструмент, позволяющий приостановить выполнение программы на определенной строке кода. Это дает возможность изучить состояние переменных и поток выполнения программы.

Как установить точку останова:

1. Откройте файл с исходным кодом.
2. Найдите строку, на которой вы хотите приостановить выполнение программы.
3. Щелкните левой кнопкой мыши в области отступа слева от этой строки или нажмите F9, чтобы установить/снять точку останова.

Типы точек останова:

- **Безусловные:** Останавливают программу всегда, когда выполняется указанная строка.
- **Условные:** Останавливают выполнение программы, только если выполнено определенное условие.
 - Щелкните правой кнопкой мыши по точке останова и выберите **Conditions**.
 - Установите условие, например $x > 10$.

Трассировка (Debugging/Stepping)

Трассировка позволяет пошагово выполнять код, чтобы отслеживать поведение программы и состояние переменных.

Основные команды трассировки:

- **F10 (Step Over):** Выполняет текущую строку и переходит к следующей, не входя внутрь вызова функции.
- **F11 (Step Into):** Входит внутрь вызываемой функции, если это возможно.
- **Shift + F11 (Step Out):** Выходит из текущей функции и возвращается к месту вызова.
- **F5 (Continue):** Продолжает выполнение программы до следующей точки останова.

Анализ содержимого переменных

Во время трассировки вы можете изучить значения переменных и объектов.

Основные способы просмотра переменных:

1. Окно локальных переменных (Locals):

- В меню выберите **Debug > Windows > Locals**.
- Отображает все локальные переменные в текущем контексте.

2. Окно автопеременных (Autos):

- В меню выберите **Debug > Windows > Autos**.
- Показывает переменные, связанные с текущей и предыдущей строкой кода.

3. Окно наблюдения (Watch):

- В меню выберите **Debug > Windows > Watch > Watch 1** (или другие).
- Позволяет вручную добавлять интересующие переменные и следить за их значениями.

4. Выражения в строке кода (Data Tip):

- Наведите курсор на переменную в коде во время трассировки, чтобы увидеть ее значение.

5. Окно быстрого просмотра (QuickWatch):

- Выделите интересующую переменную и нажмите Shift + F9, чтобы открыть окно быстрого просмотра.

Отладка №1: вычисление разности двух целых чисел на ассемблере.

Первое число: 888

Второе число: 111

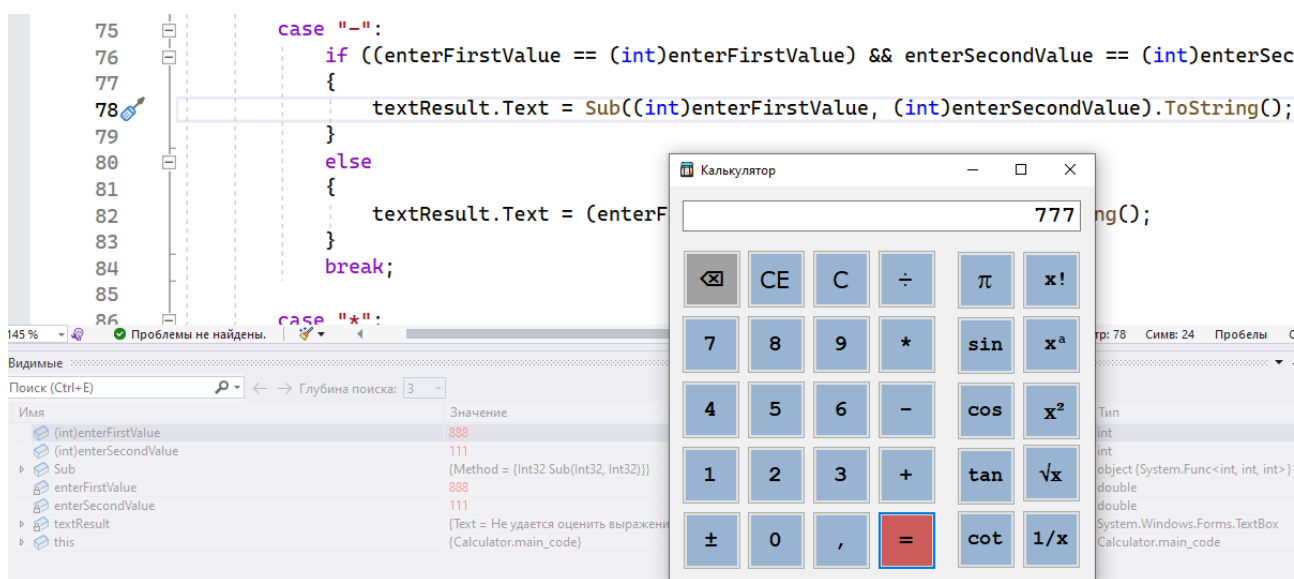


Рисунок 2 – отладка №1

Результат: 777 (рисунок 2)

Отладка №2: вычисление частного от деления двух целых чисел на ассемблере.

Делимое: 121

Делитель: 2

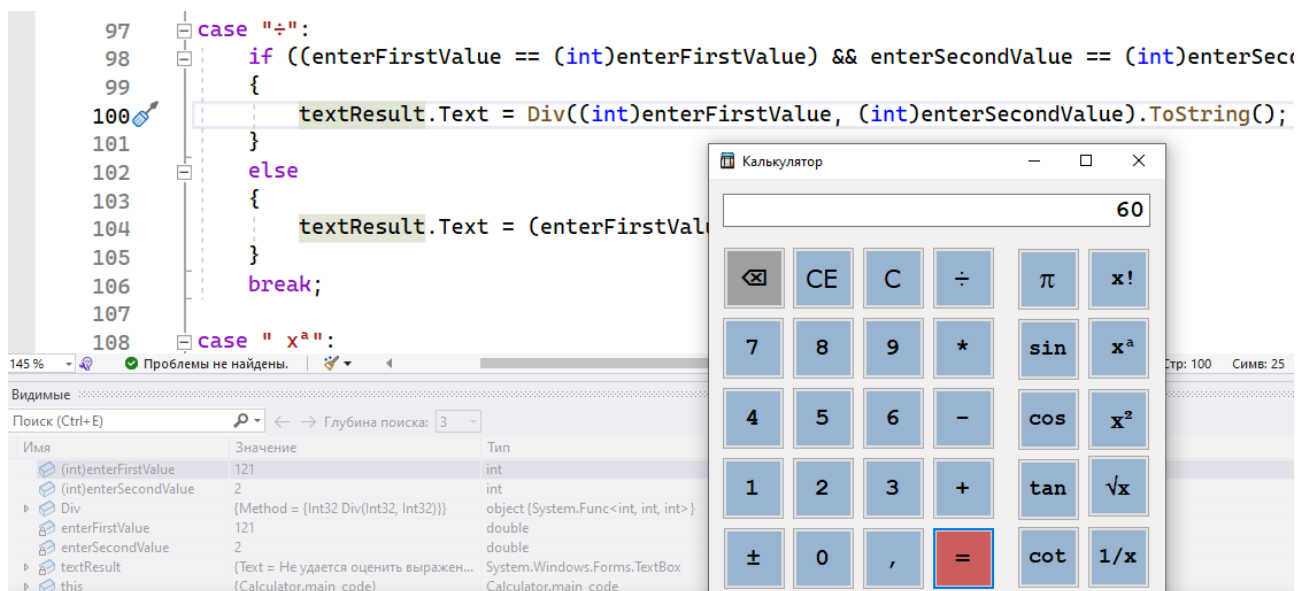


Рисунок 3 – отладка №2

Результат: 60 (рисунок 3)

Так как деление выполнялось на языке ассемблера, результатом стала только целая часть от частного.

Результаты обоих вычислений были автоматически записаны в файл calculation_history.txt (рисунок 4).

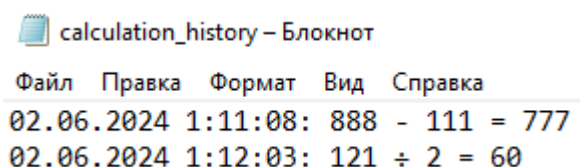


Рисунок 4 – результаты отладки в файле calculation_history.txt

4. Описание программы

Приложение Calculator.exe является основным модулем программы. При запуске программы происходит появление заставки по центру экрана, которая длится несколько секунд. Заставка состоит из картинки на заднем фоне размером 620x300 пикселей и шкалы загрузки, расположенной внизу по вертикали и посередине по горизонтали относительно заставки. Шкала загрузки не отображает реальный процесс

загрузки калькулятора и носит исключительно декоративный характер. Заставку выполняет файл splashscreen.cs, а её вид определяет файл splashscreen.Designer.cs.

После того, как заставка закончилась, открывается окно с калькулятором. Вводимые и выводимые числа в текстовом поле представлены в формате строки. Чтобы перевести их в число, используется встроенная функция Convert.ToDouble(). Первый и второй операнд записываются соответственно в переменные enterFirstValue и enterSecondValue типа double. Если была выбрана операция сложения, вычитания, умножения или деления, то при нажатии на кнопку «=» происходит проверка на целостность обоих операндов. Если оба операнда целые, то вышеописанные операции будут выполняться при помощи соответствующих функций из dll-библиотеки ASM_MATH.dll. В этой библиотеке в каждой из четырёх функций присутствует ассемблерная вставка, в которой происходит соответствующее вычисление. Если хотя бы один из операндов дробный, то вычисление будет выполнять файл main_code.cs. После вычислений результат конвертируется в строку при помощи встроенной функции Convert.ToString() и выводится на экран, а также записывается в файл calculation_history.txt. Визуальную составляющую калькулятора определяет файл main_code.Designer.cs.

Далее на рисунке 5 представлена схема взаимодействия программы.

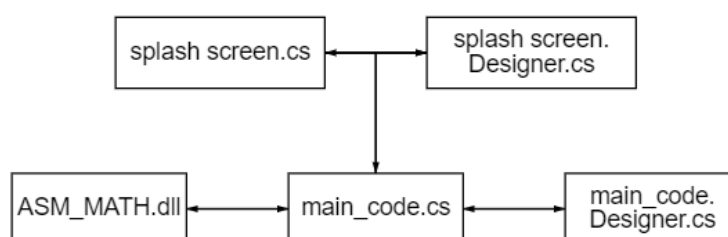


Рисунок 5 - схема взаимодействия программы

Ниже, на рисунке 6 представлена схема данных.



Рисунок 6 - схема данных

Схема программы. Функция EnterNumbers (рисунок 7).

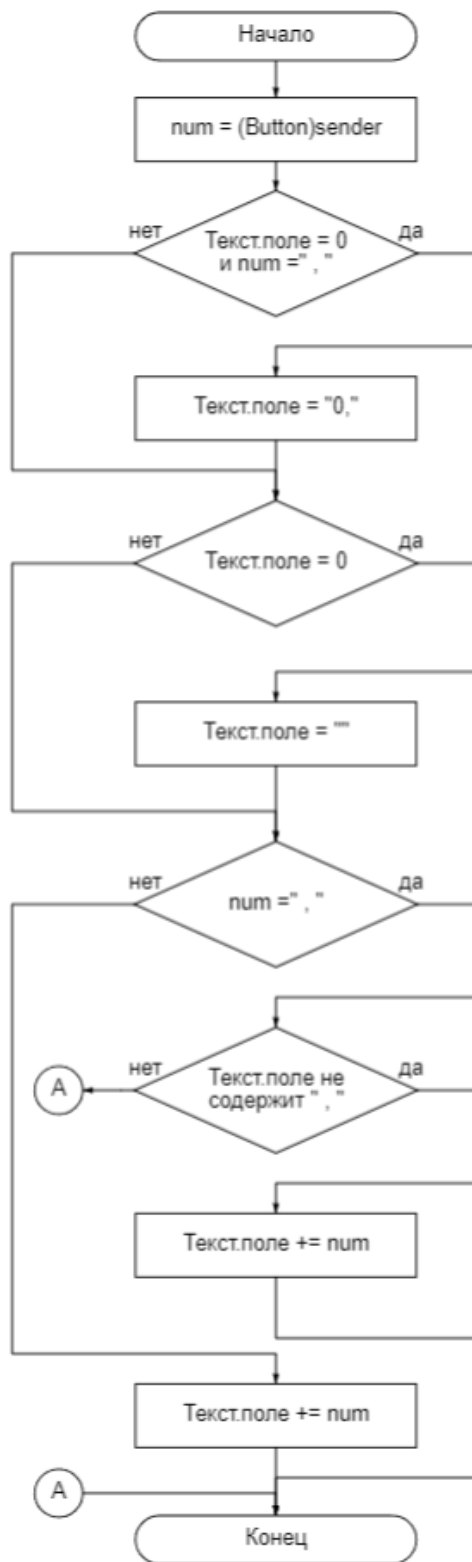


Рисунок 7 – функция EnterNumbers

Схема программы. Функция numderOper (рисунок 8).

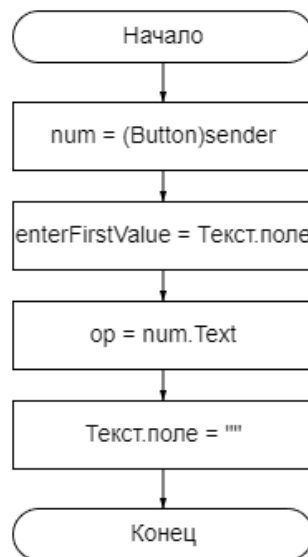


Рисунок 8 – функция numberOper

Схема программы. Функция btnEquals (рисунок 9).

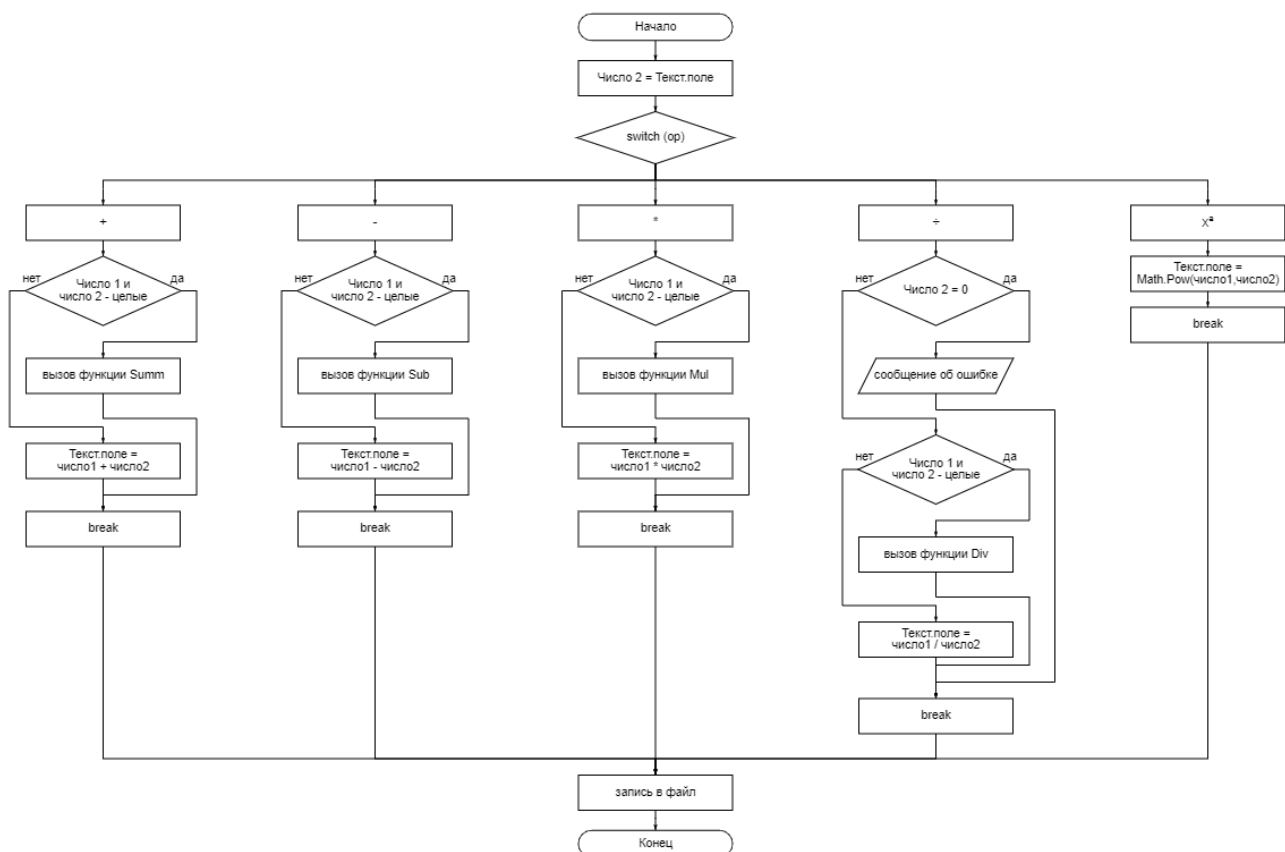


Рисунок 9 – функция btnEquals

Схема программы. Функция buttonChangeSign (рисунок 10).



Рисунок 10 – функция buttonChangeSign

Схема программы. Функция btnBS (рисунок 11).



Рисунок 11 – функция btnBS

Схема программы. Функция btnSqrt (рисунок 12).

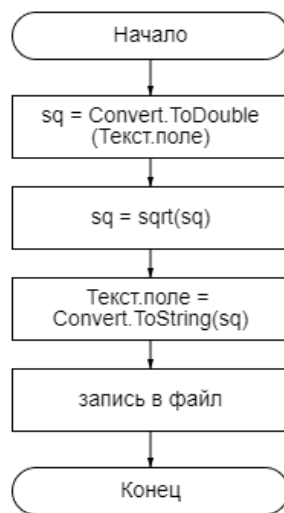


Рисунок 12 – функция btnSqrt

Схема программы. Функция `btnx2` (рисунок 13).

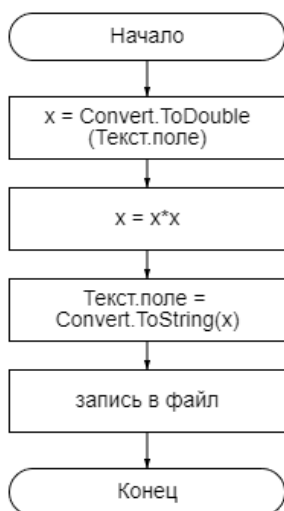


Рисунок 13 – функция btnx2

Схема программы. Функция `btn1x` (рисунок 14).

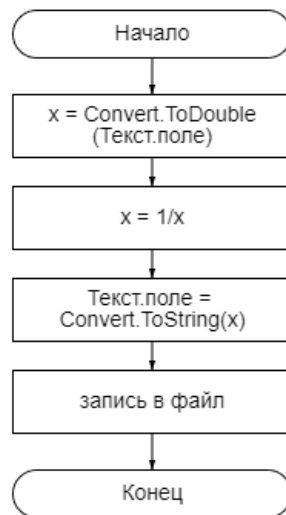


Рисунок 14 – функция btn1x

Схема программы. Функция btnSin (рисунок 15).

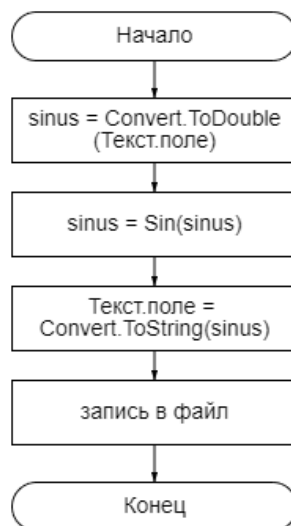


Рисунок 15 – функция btnSin

Схема программы. Функция btnCos (рисунок 16).

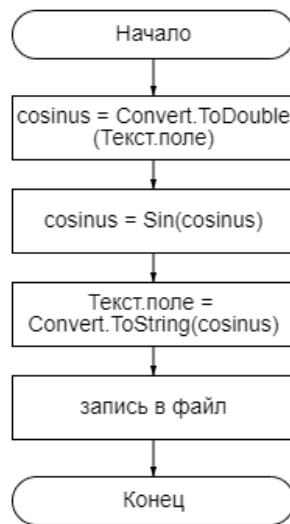


Рисунок 16 – функция btnCos

Схема программы. Функция btnTan (рисунок 17).

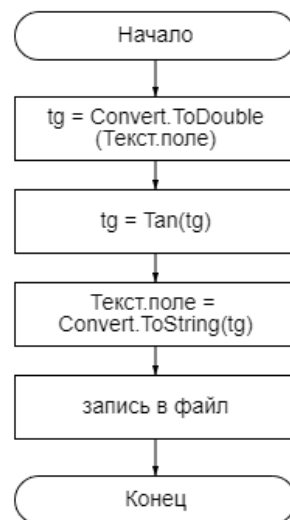


Рисунок 17 – функция btnTan

Схема программы. Функция btnCot (рисунок 18).

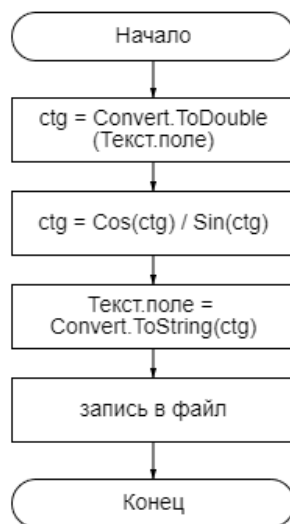


Рисунок 18 – функция btnCot

Схема программы. Функция Factorial (рисунок 19).

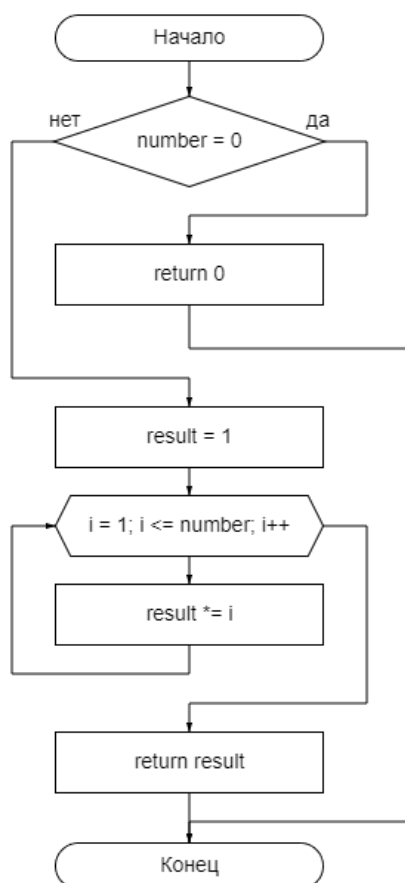


Рисунок 19 – функция Factorial

Схема программы. Функция btnFact (рисунок 20).

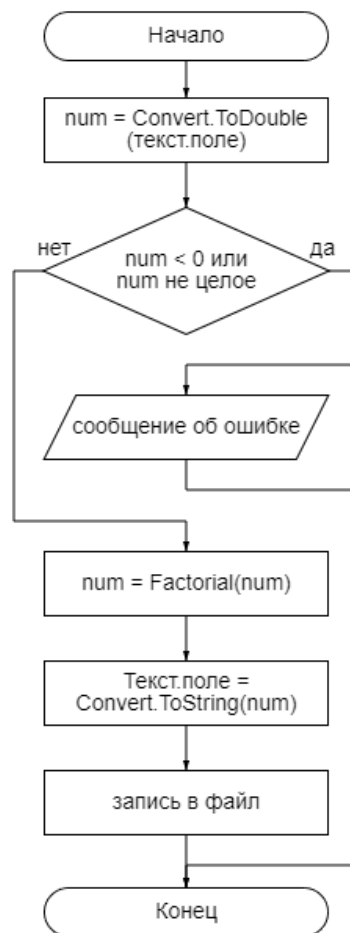


Рисунок 20 – функция btnFact

Схема программы. Функция Summ (рисунок 21).

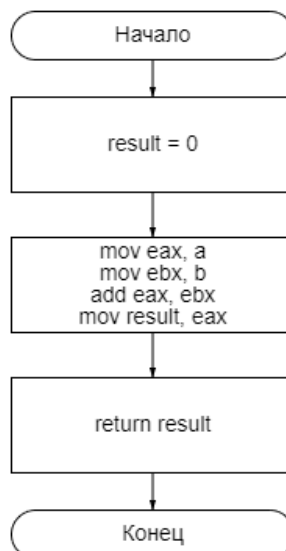


Рисунок 21 – функция Summ

Схема программы. Функция Sub (рисунок 22).

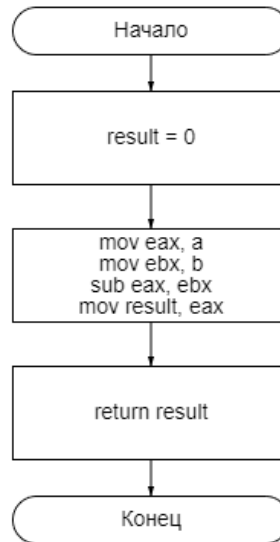


Рисунок 22 – функция Sub

Схема программы. Функция Mul (рисунок 23).

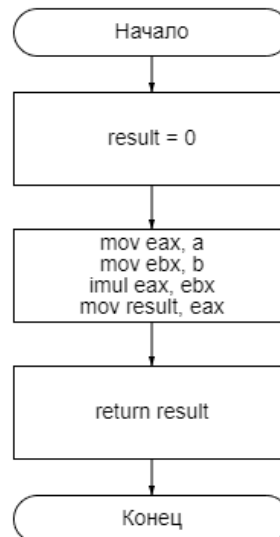


Рисунок 23 – функция Mul

Схема программы. Функция Div (рисунок 24).

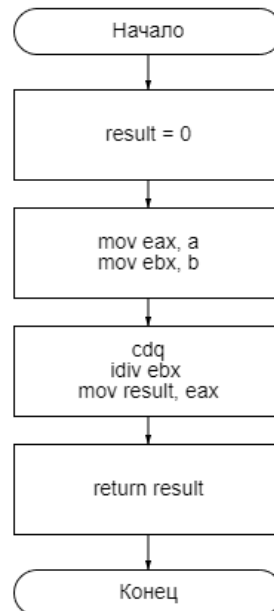


Рисунок 24 – функция Div

Ниже приведена иерархическая схема программы (рисунок 25).

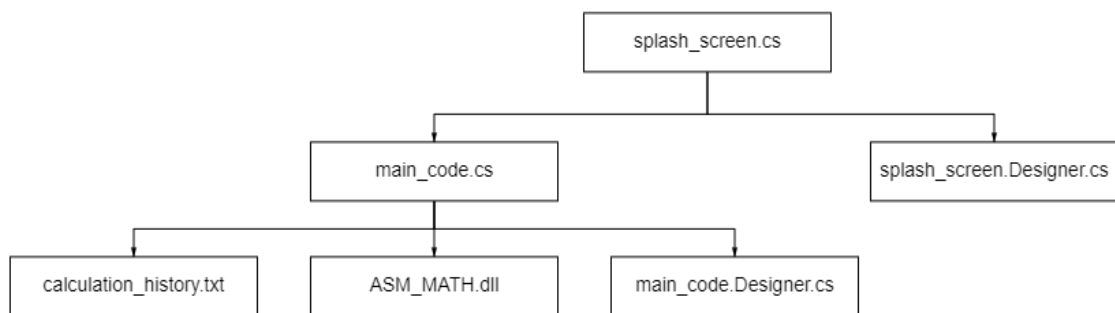


Рисунок 25 - иерархическая схема программы

5. Руководство пользователя

Программа Calculator.exe предназначена для выполнения различных математических расчётов, а также имеет интуитивно понятный интерфейс. История всех вычислений автоматически записывается в файл calculation_history.txt, который будет создан в папке с калькулятором.

5.1 Интерфейс калькулятора

- **Текстовое поле:** отображает вводимые и вычисленные значения. Числа также можно ввести с помощью клавиатуры.
- **Кнопки:**

1. **Цифры (0-9):** для ввода чисел.
2. **Запятая (","):** добавляет десятичный разделитель к числу.
3. **Стереть последний символ («»):** удаляет последний введенный символ.
4. **Удалить отображаемое число («CE»):** очищает текущее значение, оставляя все остальные введенные данные.
5. **Удалить всё («C»):** сбрасывает все введенные данные.
6. **Изменить знак («±»):** инвертирует знак текущего числа.
7. **Вычислить («=»):** выполняет вычисление на основе введенных данных.
8. **Сложение («+»):** добавляет операцию сложения.
9. **Вычитание («-»):** добавляет операцию вычитания.
10. **Умножение («*»):** добавляет операцию умножения.
11. **Деление («÷»):** добавляет операцию деления.
12. **Число Пи («π»):** вводит значение π в текстовое поле.
13. **Синус («sin»):** вычисляет синус текущего значения.
14. **Косинус («cos»):** вычисляет косинус текущего значения.
15. **Тангенс («tan»):** вычисляет тангенс текущего значения.
16. **Котангенс («cot»):** вычисляет котангенс текущего значения.
17. **Факториал («x!»):** вычисляет факториал текущего значения.
18. **Возведение в произвольную степень («x^a»):** возводит текущее значение в указанную степень.
19. **Возведение в квадрат («x²»):** возводит текущее значение в квадрат.
20. **Извлечение квадратного корня («√x»):** вычисляет квадратный корень из текущего значения.

21. Возведение в степень -1 («1/x»): вычисляет обратное значение текущего числа.

5.2 Подробное описание функций

1. Цифры 0-9

Используйте кнопки с цифрами для ввода чисел. Нажмите соответствующую кнопку, чтобы ввести цифру в текстовое поле.

2. Запятая (",")

Добавляет десятичную запятую к числу. Используйте эту кнопку, чтобы вводить дробные числа.

3. Стереть последний символ («»)

Удаляет последний введенный символ в текстовом поле. Нажмите эту кнопку, чтобы исправить неверно введенный символ.

4. Удалить отображаемое число («CE»)

Очищает текущее значение в текстовом поле, но не сбрасывает всю операцию.

5. Удалить всё («C»)

Сбрасывает все введенные данные и очищает текстовое поле.

6. Изменить знак («±»)

Инвертирует знак текущего числа (положительный на отрицательный и наоборот).

7. Вычислить («=»)

Выполняет вычисление на основе введенных данных и отображает результат в текстовом поле.

8. Сложение («+»)

Добавляет операцию сложения между двумя введенными числами.

Пример использования:

1. Введите число.
2. Нажмите кнопку «+».
3. Введите следующее число.
4. Нажмите «=» для получения результата.

9. Вычитание («-»)

Добавляет операцию вычитания между двумя введёнными числами.

Пример использования:

1. Введите число.
2. Нажмите кнопку «-».
3. Введите следующее число.
4. Нажмите «=» для получения результата.

10. Умножение («*»)

Добавляет операцию умножения между двумя введёнными числами.

Пример использования:

1. Введите число.
2. Нажмите кнопку «*».
3. Введите следующее число.
4. Нажмите «=» для получения результата.

11. Деление («÷»)

Добавляет операцию деления между двумя введёнными числами. Пример использования:

1. Введите число.
2. Нажмите кнопку «÷».
3. Введите следующее число.

4. Нажмите « $\Rightarrow$ » для получения результата.

12. Число Пи (« π »)

Вводит значение числа π (3.141592653589793). Пример использования:

1. Нажмите кнопку « π ».
2. Используйте значение π в расчетах, как любое другое число.

13. Синус («sin»)

Вычисляет синус текущего значения. Пример использования:

1. Введите число в радианах.
2. Нажмите кнопку «sin».
3. Отобразится значение синуса.

14. Косинус («cos»)

Вычисляет косинус текущего значения. Пример использования:

1. Введите число в радианах.
2. Нажмите кнопку «cos».
3. Отобразится значение косинуса.

15. Тангенс («tan»)

Вычисляет тангенс текущего значения. Пример использования:

1. Введите число в радианах.
2. Нажмите кнопку «tan».
3. Отобразится значение тангенса.

16. Котангенс («cot»)

Вычисляет котангенс текущего значения. Пример использования:

1. Введите число в радианах.

2. Нажмите кнопку «cot».
3. Отобразится значение котангенса.

17. Факториал («x!»)

Вычисляет факториал текущего значения. Пример использования:

1. Введите целое число.
2. Нажмите кнопку «x!».
3. Отобразится значение факториала.

18. Возведение в произвольную степень («x^a»)

Возводит текущее значение в указанную степень. Пример использования:

1. Введите основание (число).
2. Нажмите кнопку «x^a».
3. Введите показатель степени.
4. Нажмите «=» для получения результата.

19. Возведение в квадрат («x²»)

Возводит текущее значение в квадрат. Пример использования:

1. Введите число.
2. Нажмите кнопку «x²».
3. Отобразится значение квадрата числа.

20. Извлечение квадратного корня («√x»)

Вычисляет квадратный корень из текущего значения. Пример использования:

1. Введите число.
2. Нажмите кнопку «√x».
3. Отобразится значение квадратного корня.

21. Возведение в степень -1 («1/x»)

Вычисляет обратное значение текущего числа. Пример использования:

1. Введите число.
2. Нажмите кнопку «1/x».
3. Отобразится обратное значение.

Для выхода из программы нужно нажать «X» в правом верхнем углу экрана калькулятора.

Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Была освоена работа с файлами, изучены принципы работы с графическим интерфейсом. Также были получены основные навыки отладки и тестирования программ и программирования в среде VisualStudio 2022 на языках C# / C++ и Assembler. В рамках курсовой работы была написана программа для выполнения различных математических расчётов.

Список используемых источников

1. Рихтер Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. (2022).
2. Прата С. Язык программирования C++. 2019 г.
3. Руслан Аблязов. Программирование на ассемблере на платформе x86- 64. 2016 г.

Приложение А

Листинги программы

Файл main_code.cs

```
using System;
using System.IO;
using System.Runtime.InteropServices;
using System.Windows.Forms;

namespace Calculator
{
    public partial class main_code : Form
    {
        double enterFirstValue, enterSecondValue;
        String op;
        const string historyFilePath = "calculation_history.txt";

        public main_code()
        {
            InitializeComponent();

            [DllImport("ASM_MATH.dll", CallingConvention =
CallingConvention.StdCall)]
            public static extern int Summ(int a, int b);
```

```

        [DllImport("ASM_MATH.dll", CallingConvention =
CallingConvention.StdCall)]
        public static extern int Sub(int a, int b);
        [DllImport("ASM_MATH.dll", CallingConvention =
CallingConvention.StdCall)]
        public static extern int Mul(int a, int b);
        [DllImport("ASM_MATH.dll", CallingConvention =
CallingConvention.StdCall)]
        public static extern int Div(int a, int b);

private void EnterNumbers(object sender, EventArgs e)
{
    Button num = (Button)sender;
    if (textResult.Text == "0" && num.Text == ",")
    {
        textResult.Text = "0,";
    }

    if (textResult.Text == "0")
        textResult.Text = "";

    if (num.Text == ",")
    {
        if (!textResult.Text.Contains(","))
            textResult.Text += num.Text;
    }
    else textResult.Text += num.Text;
}

private void numderOper(object sender, EventArgs e)
{
    Button num = (Button)sender;
    enterFirstValue = Convert.ToDouble(textResult.Text);
    op = num.Text;
    textResult.Text = "";
}

public static long Factorial(double number)
{
    if (number == 0)
    {
        return 0;
    }

    long result = 1;
    for (int i = 1; i <= number; i++)
    {
        result *= i;
    }
    return result;
}

private void btnEquals_Click(object sender, EventArgs e)
{
    enterSecondValue = Convert.ToDouble(textResult.Text);

```

```

        string result = "";

        switch (op)
        {
            case "+":
                if ((enterFirstValue == (int)enterFirstValue)
&& enterSecondValue == (int)enterSecondValue)
                {
                    result = Summ((int)enterFirstValue,
(int)enterSecondValue).ToString();
                }
                else
                {
                    result = (enterFirstValue +
enterSecondValue).ToString();
                }
                break;

            case "-":
                if ((enterFirstValue == (int)enterFirstValue)
&& enterSecondValue == (int)enterSecondValue)
                {
                    result = Sub((int)enterFirstValue,
(int)enterSecondValue).ToString();
                }
                else
                {
                    result = (enterFirstValue -
enterSecondValue).ToString();
                }
                break;

            case "*":
                if ((enterFirstValue == (int)enterFirstValue)
&& enterSecondValue == (int)enterSecondValue)
                {
                    result = Mul((int)enterFirstValue,
(int)enterSecondValue).ToString();
                }
                else
                {
                    result = (enterFirstValue *
enterSecondValue).ToString();
                }
                break;

            case "÷":
                if (enterSecondValue == 0)
                {
                    MessageBox.Show($"Нельзя делить на 0.");
                    break;
                }

```

```

    }

    if ((enterFirstValue == (int)enterFirstValue)
&& enterSecondValue == (int)enterSecondValue)
    {
        result = Div((int)enterFirstValue,
(int)enterSecondValue).ToString();
    }
    else
    {
        result = (enterFirstValue /
enterSecondValue).ToString();
    }
    break;

    case " xa":
        result = (Math.Pow(enterFirstValue,
enterSecondValue).ToString());
        break;

    default:
        break;
}

textResult.Text = result;
if (op == " xa")
{
    LogCalculation($"{enterFirstValue}^{(enterSecondValue)} = {result}");
}
else LogCalculation($"{enterFirstValue} {op}
{enterSecondValue} = {result}");
}

private void LogCalculation(string calculation)
{
    try
    {
        using (StreamWriter writer = new
StreamWriter(historyFilePath, true))
        {
            writer.WriteLine($"{DateTime.Now}:
{calculation}");
        }
    }
    catch (IOException ex)
    {
        MessageBox.Show($"Ошибка при записи в файл:
{ex.Message}");
    }
}

private void buttonChangeSign_Click(object sender,
EventArgs e)

```

```

    {
        if (textResult.Text[0] == '-')
            textResult.Text = textResult.Text.Remove(0, 1);
        else
            textResult.Text = "-" + textResult.Text;
    }

    private void ClearEntry_Click(object sender, EventArgs e)
    {
        textResult.Text = "0";
    }

    private void btnClear_Click(object sender, EventArgs e)
    {
        textResult.Text = "0";
        enterFirstValue = 0;
    }

    private void btnBS_Click(object sender, EventArgs e)
    {
        if (textResult.Text.Length > 0)
        {
            textResult.Text =
textResult.Text.Remove(textResult.Text.Length - 1, 1);
        }

        if (textResult.Text == "")
        {
            textResult.Text = "0";
        }
    }

    private void btnPi_Click(object sender, EventArgs e)
    {
        textResult.Text = "3,14159265358979323846264";
    }

    private void btnSqrt_Click(object sender, EventArgs e)
    {
        double sq = Convert.ToDouble(textResult.Text);
        sq = Math.Sqrt(sq);
        LogCalculation($"sqrt({textResult.Text}) = {sq}");
        textResult.Text = Convert.ToString(sq);
    }

    private void btnx2_Click(object sender, EventArgs e)
    {
        double x;
        x = Math.Pow(Convert.ToDouble(textResult.Text), 2);
        LogCalculation($"{{textResult.Text}}^2 = {x}");
        textResult.Text = Convert.ToString(x);
    }

    private void btn1x_Click(object sender, EventArgs e)

```

```

    {
        double x;
        x = Math.Pow(Convert.ToDouble(textResult.Text), -1);
        LogCalculation($"1/{textResult.Text} = {x}");
        textResult.Text = Convert.ToString(x);
    }

private void btnSin_Click(object sender, EventArgs e)
{
    double sinus = Convert.ToDouble(textResult.Text);
    sinus = Math.Sin(sinus);
    LogCalculation($"sin({textResult.Text}) = {sinus}");
    textResult.Text = Convert.ToString(sinus);
}

private void btnCos_Click(object sender, EventArgs e)
{
    double cosinus = Convert.ToDouble(textResult.Text);
    cosinus = Math.Cos(cosinus);
    LogCalculation($"cos({textResult.Text}) = {cosinus}");
    textResult.Text = Convert.ToString(cosinus);
}

private void btnTan_Click(object sender, EventArgs e)
{
    double tg = Convert.ToDouble(textResult.Text);
    tg = Math.Tan(tg);
    LogCalculation($"tan({textResult.Text}) = {tg}");
    textResult.Text = Convert.ToString(tg);
}

private void btnCot_Click(object sender, EventArgs e)
{
    double ctg = Convert.ToDouble(textResult.Text);
    ctg = Math.Cos(ctg) / Math.Sin(ctg);
    LogCalculation($"cot({textResult.Text}) = {ctg}");
    textResult.Text = Convert.ToString(ctg);
}

private void btnFact_Click(object sender, EventArgs e)
{
    double num = Convert.ToDouble(textResult.Text);
    if (num < 0 || num != (int)num)
    {
        textResult.Text = "Error";
    }
    else
    {
        textResult.Text = Convert.ToString(Factorial(num));
        LogCalculation($"{num}! = {textResult.Text}");
    }
}
}
}

```

Файл splash_screen.cs

```
using System;
using System.Windows.Forms;

namespace Calculator
{
    public partial class splash_screen : Form
    {
        public splash_screen()
        {
            InitializeComponent();

            private void timer1_Tick(object sender, EventArgs e)
            {
                timer1.Enabled = true;
                progressBar1.Increment(3);
                if (progressBar1.Value == 100)
                {
                    timer1.Enabled = false;
                    main_code form = new main_code();
                    form.Show();
                    this.Hide();
                }
            }
        }
    }
}
```

Файл ASM_MATH.dll

```
#include "pch.h"

BOOL APIENTRY DllMain( HMODULE hModule,
                      DWORD ul_reason_for_call,
                      LPVOID lpReserved
                      )
{
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}

#include <utility>
#include <limits.h>
#include "ASM_MATH.h"

int Summ(int a, int b) {
```

```

    int result = 0;
    _asm {
        mov eax, a
        mov ebx, b
        add eax, ebx
        mov result, eax
    }
    return result;
}

int Sub(int a, int b) {
    int result = 0;
    _asm {
        mov eax, a
        mov ebx, b
        sub eax, ebx
        mov result, eax
    }
    return result;
}

int Mul(int a, int b) {
    int result = 0;
    _asm {
        mov eax, a
        mov ebx, b
        imul eax, ebx
        mov result, eax
    }
    return result;
}

int Div(int a, int b) {
    int result = 0;
    _asm {
        mov eax, a
        mov ebx, b
        cdq
        idiv ebx
        mov result, eax
    }
    return result;
}

```

Приложение В

Результат работы программы

Приложение В.1 – Заставка

Инженерный калькулятор

Разработчик:
Пырков Денис

Группа:
23ВВВ2



Приложение В.2 – Окно калькулятора

