Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

По курсу «Программирование»

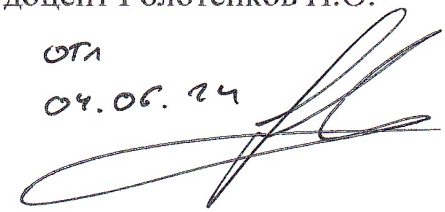
На тему «Калькулятор»

Выполнил: студент группы 23ВВВ4:

Фельдман Г.О.

Принял:

к.т.н доцент Голотенков Н.О.



Пенза 2024

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра “Вычислительная техника”

“УТВЕРЖДАЮ”

Зав. кафедрой ВТ

_____ М.А. Митрохин

«____»_____2024 г.

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Фельдман Глебу Олеговичу Группа 23ВВВ4

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

Исходные данные (технические требования) на проектирование

Разработка программы приложения – «инженерный калькулятор». Программа должна иметь текстовый или графический интерфейс, выполнять основные арифметические операции такие как: сложение, вычитание, умножение, деление, возведение в степень, извлечение квадратного корня, вычислять тригонометрические выражения (sin, cos, tg, ctg), переводить десятичные числа в дополнительный код. Также результат текущих вычислений программа должна вычислять в файл.

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 "09" февраля 2024 г.

Дата защиты проекта "04" 06. 24.

Руководитель Голотенков Николай Олегович

Задание получил "09" февраля 2024 г.

Студент Фельдман Глеб Олегович

Оглавление

Введение..... **5**
Постановка задачи **6**
Выбор решения **6**
Описание разработки программы **7**
Отладка и тестирование..... **9**
Описание программы **12**
Руководство пользователя **15**
Заключение **19**
Список используемых источников **20**

Введение

В современном мире программирование является одной из самых востребованных областей среди специалистов информационных технологий. Каждый день программисты создают новые программы и приложения, упрощающие жизнь людей. Одним из таких полезных инструментов является калькулятор. Во многом благодаря такому продукту люди могут сократить время выполнения той или иной задачи, которая требует значительные усилия в вычислениях.

В рамках данной курсовой работы будет рассмотрено создание калькулятора с использованием языков программирования C/C++ и Assembler. Калькулятор - инструмент, который позволяет выполнять различные математические операции, а также использовать различные функции, такие как косинус, синус, тангенс. Будет реализован подсчет корня, степени и экспоненты. Все вычисляемые выражения будут сохраняться в файл

В процессе создания калькулятора будут использованы основные принципы программирования, такие как работа с переменными, операторами, условными операторами, циклами и функциями.

Целью данной работы является изучение основных принципов создания калькулятора и развитие навыков программирования. В ходе выполнения курсовой работы будет создан функциональный калькулятор, который сможет выполнять основные математические операции и функции.

Таким образом, изучение создания калькулятора позволит углубить свои знания в программировании, а также научиться применять их на практике для разработки полезных программных инструментов.

Результатом же станет программа, которая исполняет все востребованные функции.

1. Постановка задачи

Необходимо разработать программу для работы с вычислениями инженерный калькулятор. Программа должна иметь текстовый список функций и операций, которые возможно реализовать при работе, для понимая рядового пользователя. Необходимо построить алгоритмы работы с такими задачами как: сложение, вычитание, умножение, деление, возведение в степень, извлечение корня, вычислять тригонометрические выражения (sin, cos, tg, ctg), переводом из десятичных чисел в дополнительный код. Также результат текущих вычислений программа должна вычислять в файл.

Многомодульность программы. Программа должна быть поделена на логические модули. Это упростит поиск ошибок при отладке и тестировании программы, а также позволит легко расширять функционал программы. Использование сложных типов данных – массивов, структур. Это необходимо для обработки данных в коде программы.

Режим работы калькулятора – текстовый.

Устройство ввода-вывода – клавиатура и мышь. Необходимо различать и идентифицировать действия, произведенные с их помощью, это облегчит использование программы.

Интерфейс должен быть построен на основе меню. Это необходимо для создания интуитивно понятного интерфейса для пользователя. Заставка необходима для того, чтобы пользователь, запустивший программу, смог получить достаточную информацию о ней.

2. Выбор решения

При запуске программы открывается приветственный экран, с кратким описанием возможных функций. Вычисление итогового значения происходит при помощи стека следующим образом

1) Считывание данных строки. На данном этапе будет считываться каждый символ строки и определяться как операция или число.

2) Расстановка приоритета. Здесь будет расставлен приоритет между операциями.

3) Занесение в стек в соответствии с приоритетом операции

4) Вычисление. На этом этапе с помощью кейсов будут произведены основные подсчеты. Для подсчетов тригонометрических и экспоненциальных выражений будем пользоваться библиотекой <math.h>

5) Занесение в стек результата и удаление произведенной операции.

6) Вывод данных. Вывод будет производиться как в консоль, так и в файл.

7) Выход из программы.

Для решения данной задачи мы будем использовать следующую структуру данных:

```
struct Leksema
{
        char type;
        double value;
};
```

Выбранная структура очень проста и соответствует всем потребностям программы. Для чисел был создан value, а для операций type соответственно.

3. Описание разработки программы

После выбранного решения и описания структуры данных, главная проблема в реализации приоритета операций. Как уже и было сказано ранее, можно использовать 2 стека, причем вносить и извлекать мы можем только с самой вершины. В стеке с операциями, каждому символу будет соответствовать число (приоритет). После занесения из строки следующей операции, должна производится проверка:

1) Если последующая операция по заданному приоритету выше, то в таком случае мы просто вносим её в стек

2) Если бы была операция схожая или ниже по приоритету, то тогда сначала выполняем предыдущую операцию

В каждом случае, достигнув конца строки, мы последовательно из стека берем пару чисел (число в случае функции) и операцию, производим вычисления этой пары, удаляем соответствующие взятые элементы из стека, результат же передаем обратно в стек для последующих вычислений.

Далее необходима проверка обоих стеков на их размер, в случае если в стеке с операциями не будет знаков, а в стеке с числами всего одно значение, то будет выведен ответ.

Обратившись к структуре, мы понимаем, что необходимо и операции, и числам задавать тип, а также значение. В случае числа, её тип мы идентифицируем как «0», а значение будет соответствовать считанному до операции. В операции же значение будет нулевым, а тип – значению первого символа, по которому мы будем обращаться в последующем при вычислениях.

Немало важным шагом в разработке инженерного калькулятора являются скобки. Скобки будут реализованы следующим образом:

Считывем открывающую скобку как операцию и заносим в стек, после чего заносим и само выражение в скобках, после того как мы встретим закрывающую скобку, мы производим математические операции внутри этих скобок. В последствии удаляем выполненное и результат заносим в стек.

Таким образом, разработанная программа состоит из нескольких модулей(функций).

1) main – основная функция. Это связующее звено остальных функций, в нем будут считываться данные.

2) asm_funk – функция для приветствия и краткой сводки возможностей программы.

3) Parara – функция для расстановки приоритетов операции.

4) Maths – функция для подсчета выражения

5) decimal_to_binary – функция для перевода чисел из десятичной формы в дополнительный код.

Реализация выполняется «сверху-вниз» (нисходящая разработка), т. е. вначале проектируем компоненты верхних уровней иерархии, затем следующих и так далее до самых нижних уровней. В той же

последовательности выполняем и реализацию компонентов.

4. 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 Visual Studio: точка останова, трассировка, анализ содержимого переменных. Тестирование проводилось во время разработки и также после завершения разработки. В ходе нее было выявлено огромное количество проблем.

Для исправления выявленных ошибок я использовал главным образом «пошаговое выполнение кода». Такие команды позволяют больше узнать о процессе и порядке выполнения программы.

Во вкладке «Отладка» существует несколько команд: «Шаг с заходом», «Шаг с обходом», «Шаг с выходом».

Шаг с заходом (клавиша F11) – используется для остановки выполнения на каждом операторе кода.

Шаг с обходом (клавиша F10) – используется для выполнения кода, в том случае, если текущая строчка содержит вызов функции, а затем останавливает выполнение после возврата управления функции.

Шаг с выходом (сочетание клавиш Shift+F11) – используется для повторного выполнения кода и его приостановки в том случае, когда текущая функция возвращает управление.

При использовании данных команд важным аспектом является «Точка останова», при её использовании (клавиша F9) можно сразу пройти отладку до необходимого места. Стоит обращать внимание на такую вкладку в отладке «Окна», как локальные переменные. Это окно выводит значения существующих на данный момент переменных в программе.

Первой ошибкой при разработке программы был вывод только последней произведенной операции. С помощью отладки, команды шага с обходом, точки останова (размещённой на возможных ошибках, где

необходимо посмотреть значения переменных) и окном с видимыми переменными удалось выявить ошибку.

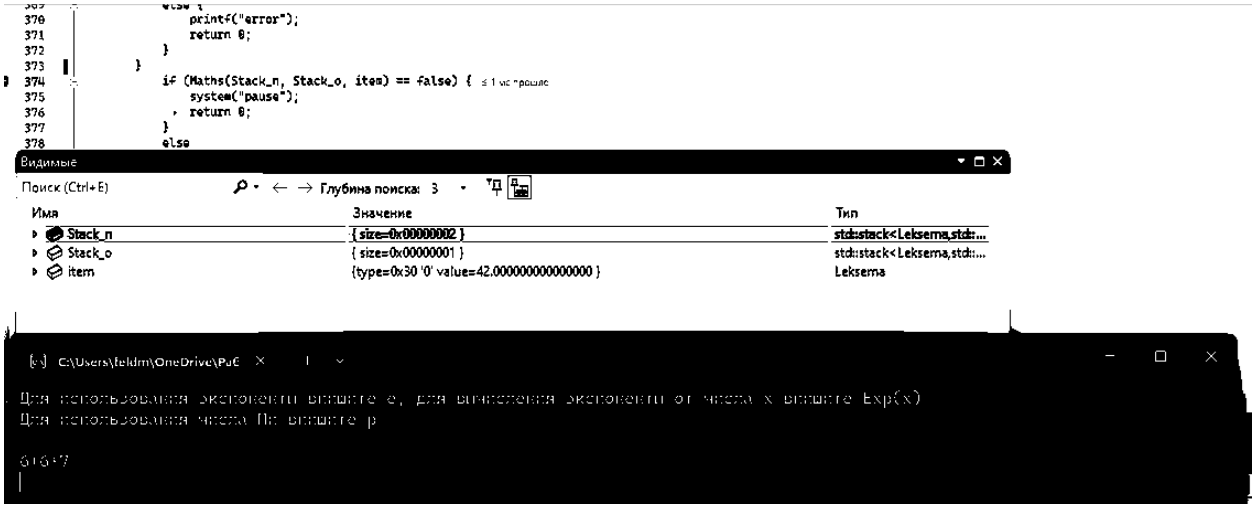


Рисунок 1 – программная ошибка №1

В локальных переменных видно, что item.value =42, что значит выполнилась последняя введенная операция. В то же время стек с операциями равен 1, хотя код после прекращает работу.

Проблема происходит из-за того, что мы вызываем математическую функцию только единожды, когда необходимо это повторять до тех пор, пока стек не будет равен 0. Таким образом решение:

Рисунок 2 – решение программной ошибки №1

Другая ошибка заключалась в том, что если записать в начало строки число с -, то программа не понимает из чего необходимо вычитать (например :-2*5).

Все дело в том, что в таком случае нам необходимо считать – в начале строки не как операцию, а как отрицательное число.

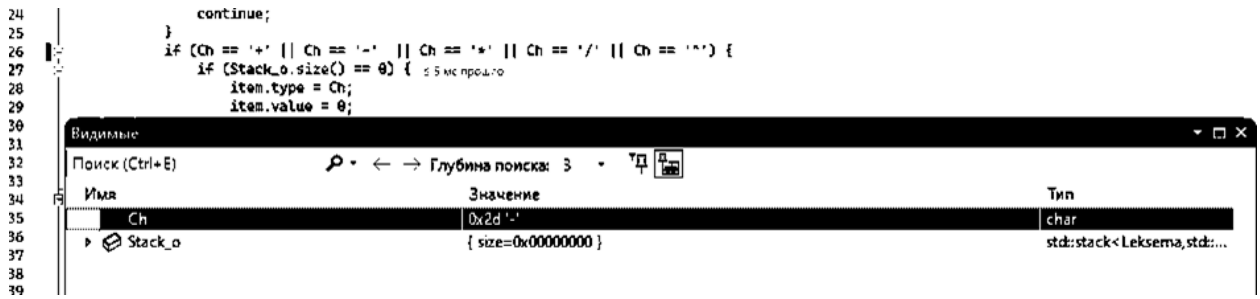


Рисунок 3 – программная ошибка №2

Для этого добавим флаг, который изначально будет равен 1(то есть будет считывать как отрицательное число) и будет изменятся в процессе прохождения условия с числами на значение 0 (в таком случае, будет восприниматься как вычитание). В условии для операций необходимо выставить «–» со значением флага соответственно.

В таком случае «–» в начале строки вычисляется корректно.

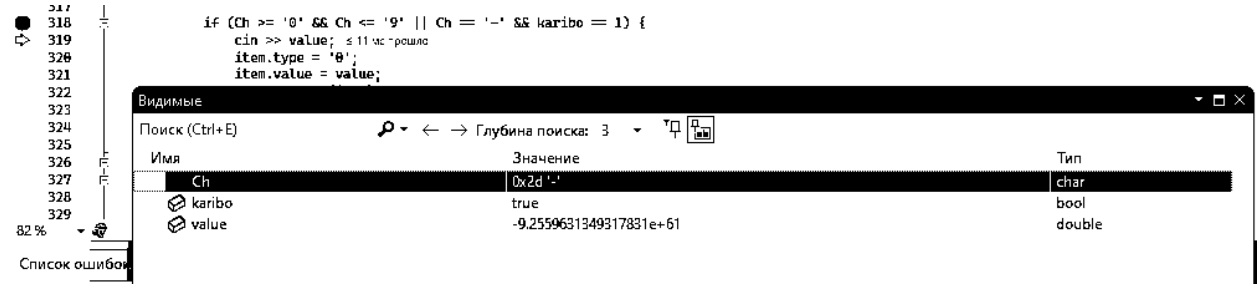


Рисунок 4 – решение программной ошибки №2

Получим вывод:

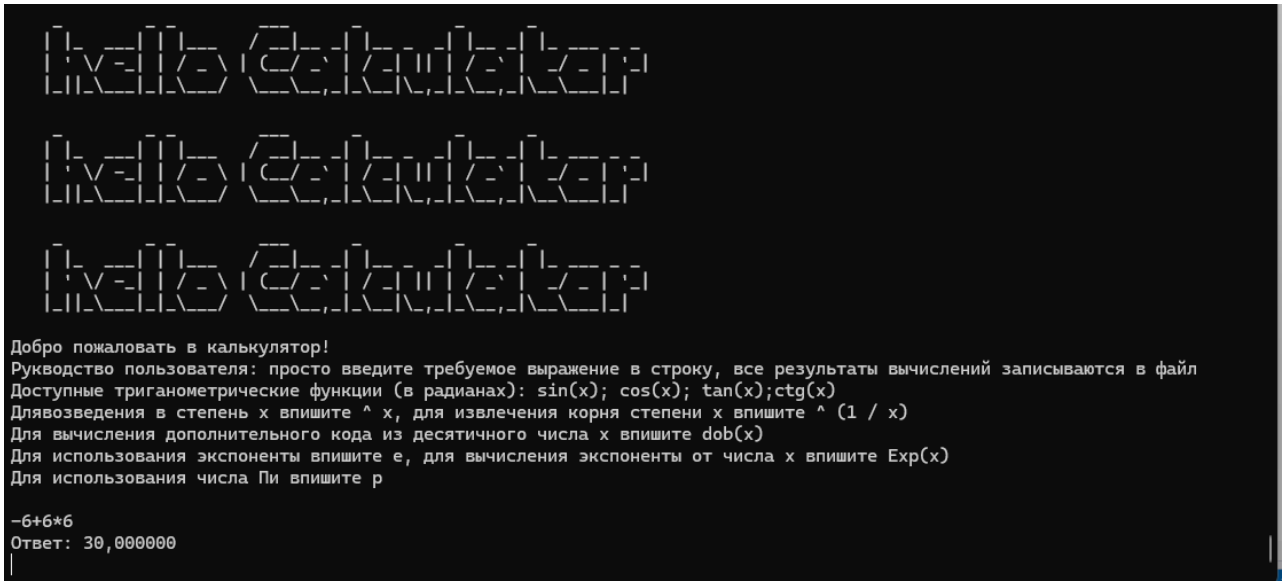


Рисунок 5 – результат вывода с исправленными ошибками

5. Описание программы

При запуске программы происходит вызов функции asm_func через ассемблер для вывода заставки-приветствия, а также руководства пользователя. После чего можно приступать непосредственно к вычислениям.

Далее на рисунке представлена схема взаимодействия программы.

Схема показывает состав модулей и управляющие отношения между ними. В ней представлен модуль наиболее высокого уровня -Main(), который взаимодействует с модулями более низкого уровня: Parara, а также Maths, возвращающие значения и asm_func, который не возвращает значений – составляющие 2 уровень. decimal_to_binary () связан и возвращает значения с Maths, таким образом являющийся модулем последнего – 3 уровня.

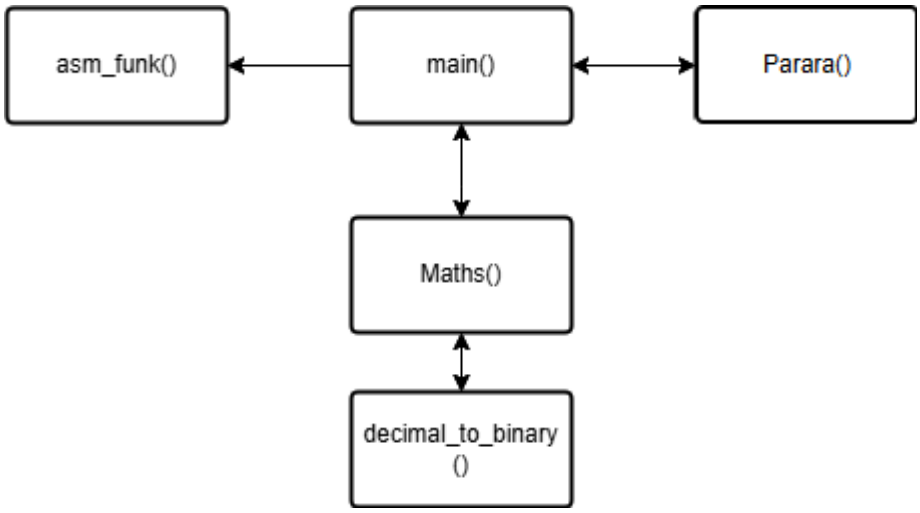


Рисунок 6 – схема взаимодействия программы

В приложении А написан подробный код программы – файла calc.cpp.

Функция main является основным модулем программы. Данная функция включает основные этапы программы: считывание символов строки, передача данных в стек, выполнение математических операций над данными и вывод приветственного экрана вызовом функции Maths и asm_func соответственно, вывод результата и его запись в файл.

Схема программы. Функция Main.

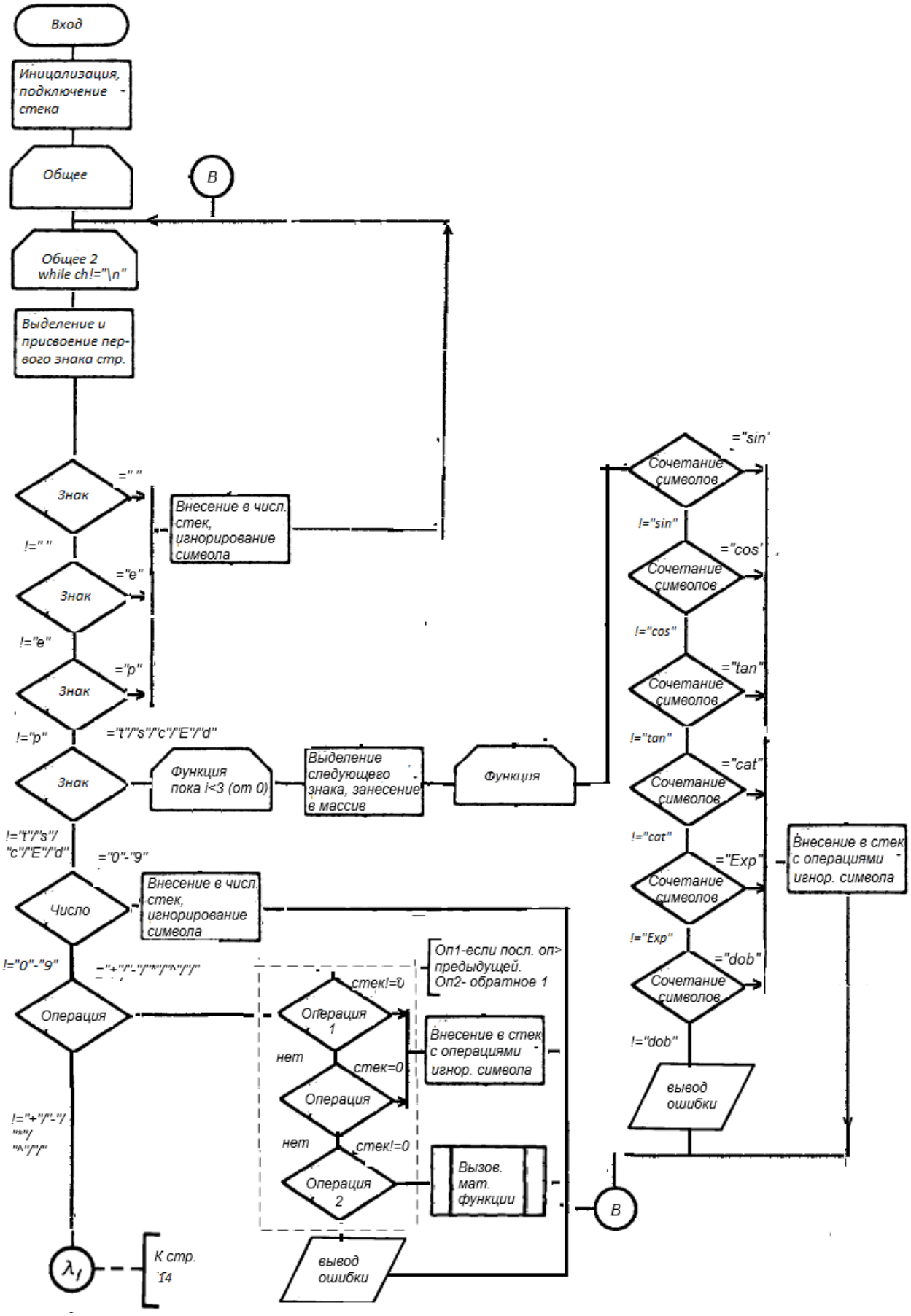


Рисунок 7.1 – схема функции main

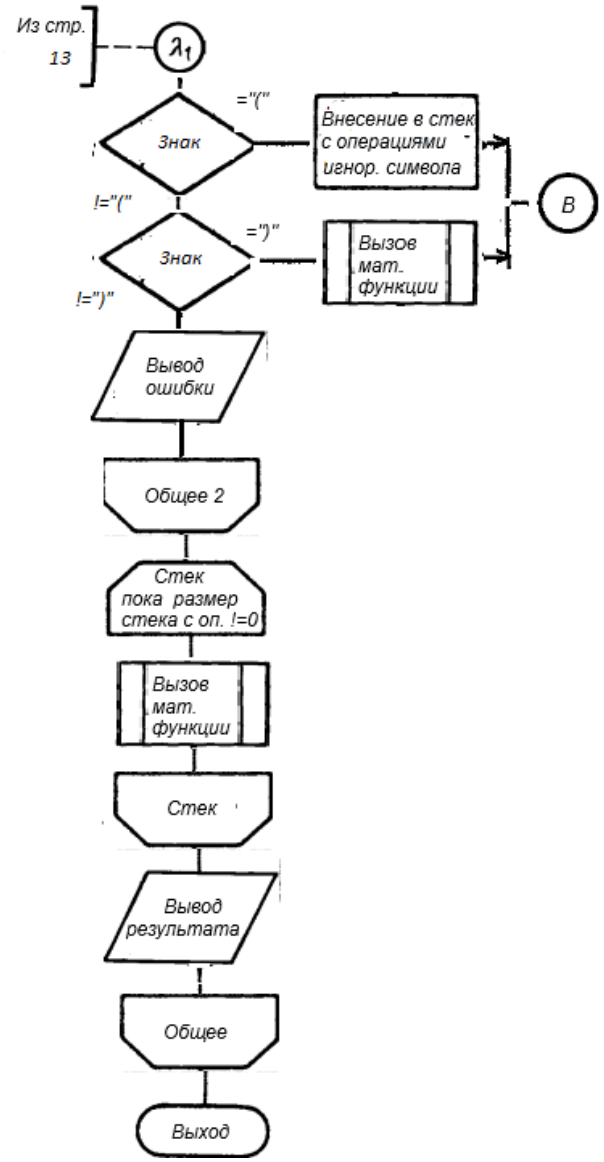


Рисунок 7.2 – схема функции main

Механизм программы событийный, большая часть считывания символов строки реализована условными операторами if, else.

Реализована функция decimal_to_binary () преобразующая десятичное число в двоичный код, или в случае ввода отрицательного числа в дополнительный.

Функция Maths же выполнена посредствам использования оператора множественного выбора switch. Каждой операции соответствует свой собственный тип, с помощью которого она будет обрабатываться.

Модуль программы Parara() присваивает операциям их приоритет, для корректного результата вычислений. Он также по большей части построен на условных операторах. Наиболее высокие по степени выполнения являются

функции, скобки и степень, далее идут умножение и деление, в самом же конце выполнения находятся сложение и деление.

6. Руководство пользователя

Программа calc.cpp предназначена для вычисления строк математических операций. Программа имеет консольный интерфейс и поддерживает такие операции как сложение, вычитание, деление, умножение, возведение в степень, извлечение корня, вычисление тригонометрических функций (синус, косинус, тангенс, котангенс), перевод десятичных чисел в двоичный (дополнительный) код. При вычислении есть возможность пользоваться экспонентой и числом пи. При этом, каждый ваш результат будет записываться в файл.

После запуска будет выведена приветственная заставка и список функций.

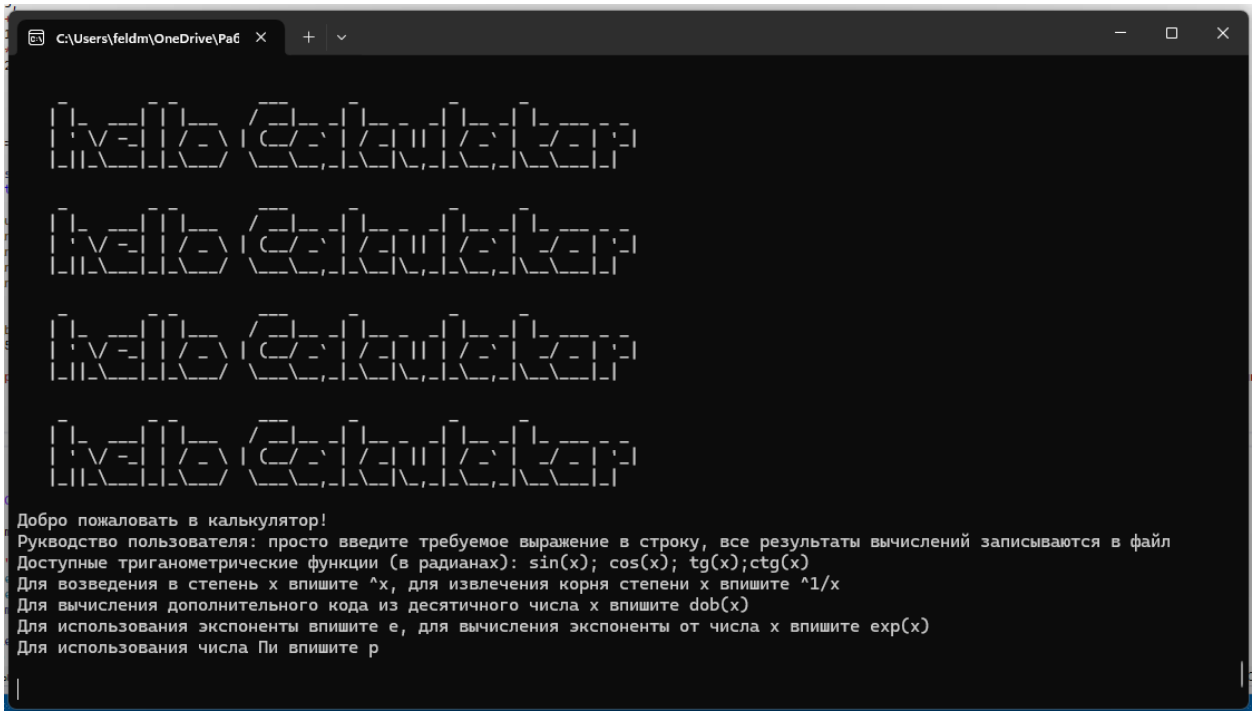


Рисунок 8 – начальный экран

Для вычисления вашего выражения, необходимо вписать его в строку, используя предложенный на экране синтаксис и нажать Enter. Далее представлен вывод простейших математических операций (+,-,/,*).

<число><операция><число>..

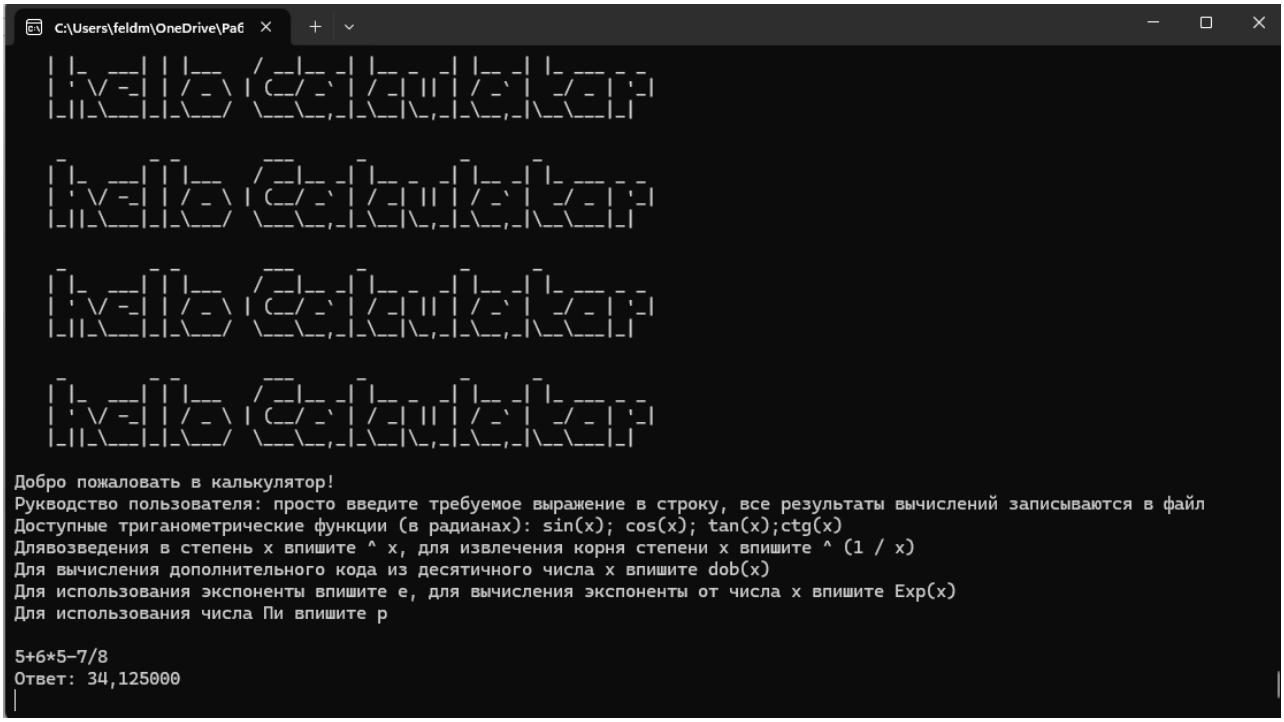


Рисунок 9 – вычисление с прост. операциями

Вы можете возвести число в степень. Аналогичный вывод будет и для констант p или e. <число>^<степень>

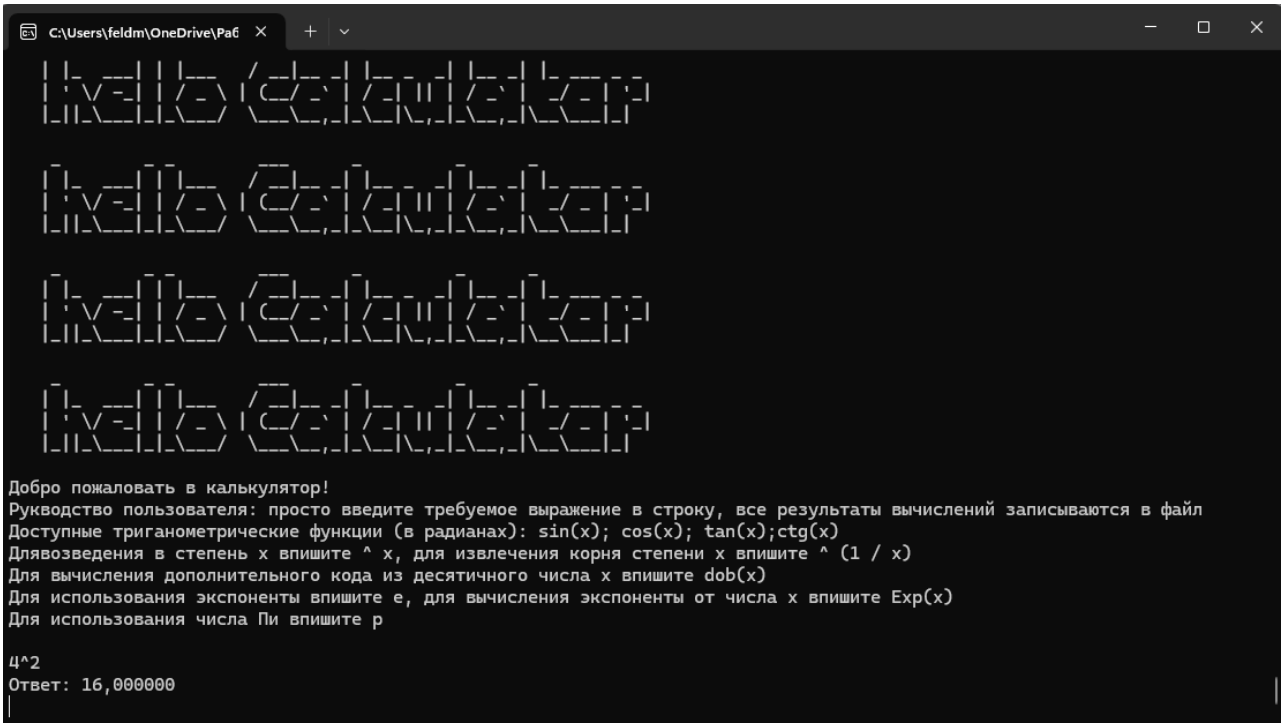


Рисунок 10 – вычисление со степенью

Для вычисления квадратного корня числа вам потребуется ввести <число>^1/<степень нужного корня>

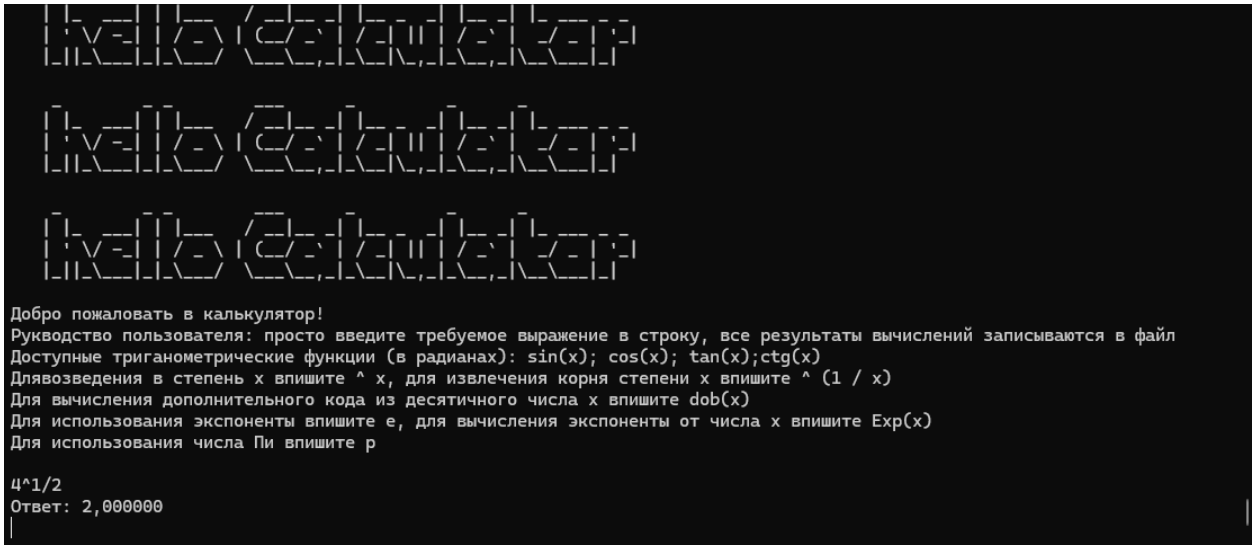


Рисунок 11 – вычисление с корнем

Важно вписывать функции со скобками, где в скобках находится нужный нам аргумент <функция>(<аргумент/число>). Далее представлен вывод вычисления функции sin (аналогично и для других функций (cos,tan,ctg).

Важно! Результат функций выводится исключительно в радианах.

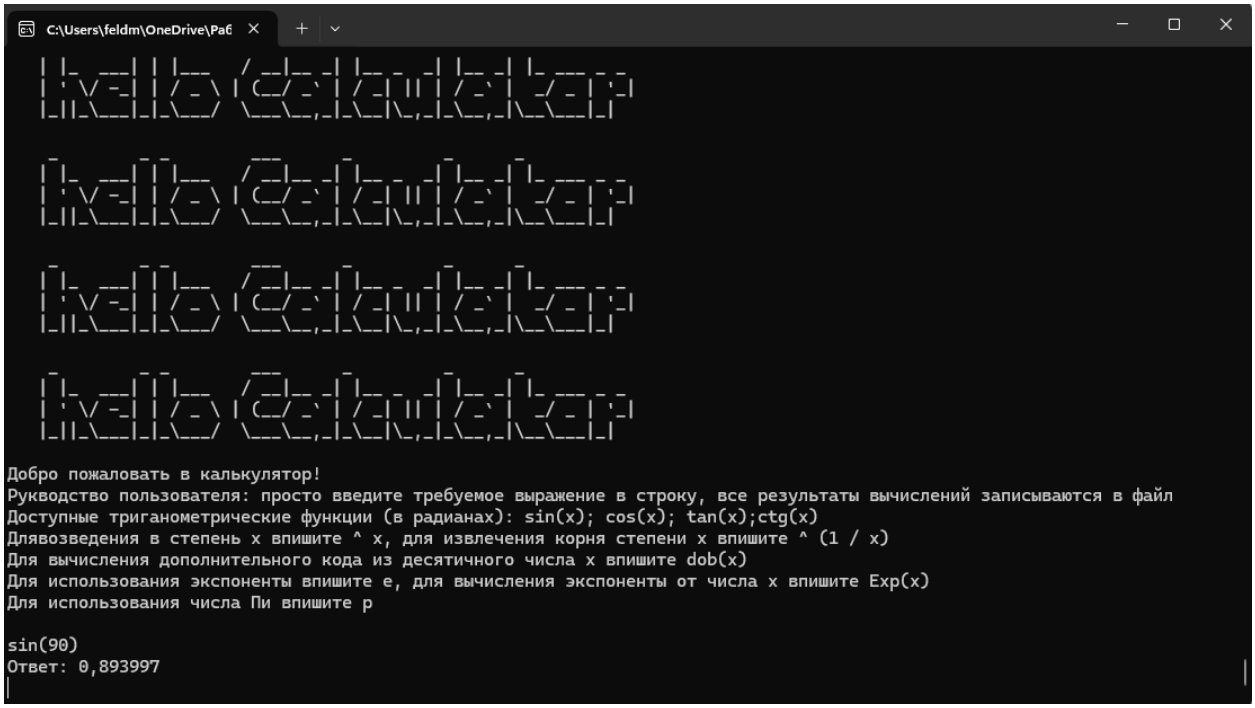


Рисунок 12 – вычисление с функцией

Для вычисления дополнительного или двоичного кода впишите сочетание dob(<число>). В случае если число отрицательное, вам необходимо добавить 1 слева для нужной вам разрядности (в противном случае 0).

Для корректности вычислений для последующего вычисления двоичного/дополнительного кода рекомендуется перезапуск программы.

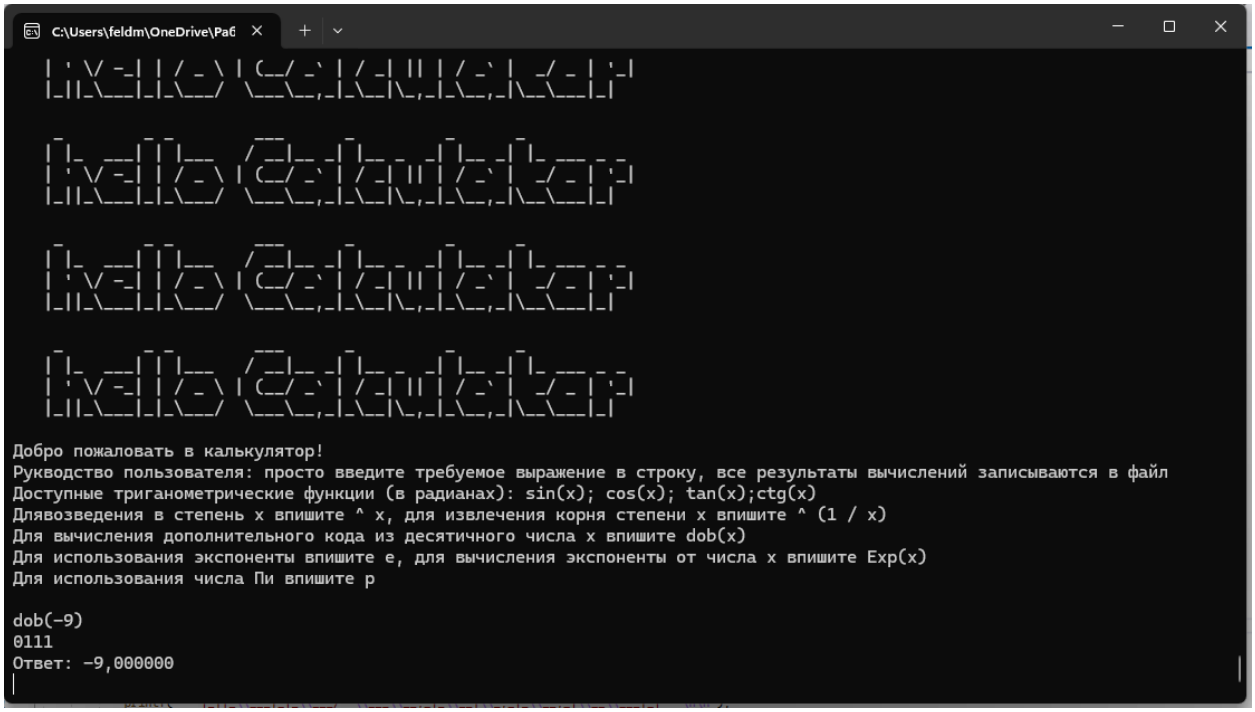


Рисунок 13 – вычисление с доп.кодом

Присутствует возможность вычисления экспоненты от числа. Exp(<число>)

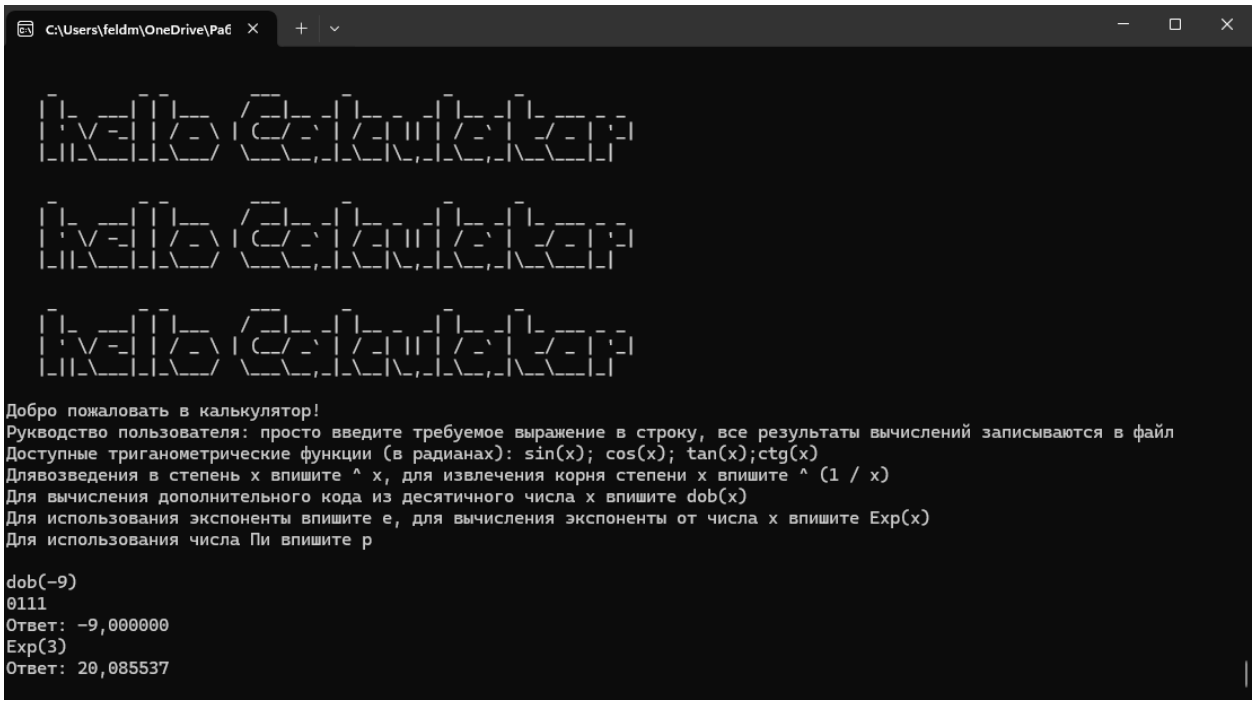


Рисунок 14 – вычисление с экспонентой

Все вычисленные данные будут последовательно сохраняться в файл. Результаты вычислений представлены в приложении С.

Заключение

При выполнении курсовой работы были получены навыки разработки многомодульных программ. Были освоены способы обработки событий при вводе с клавиатуры и мыши. Усвоены базовые навыки программирования на языке C/ C++ и Ассемблер. Изучены основные возможности среды программирования Visual Studio 2022. Получены навыки тестирования и отладки программ.

В рамках выполнения курсовой работы была написана программа, необходимая для вычисления математических выражений. Программа предоставляет небольшой, но достаточный для использования список возможностей. Не требует большого количества системных ресурсов и обеспечивает быструю обработку строки выражений.

В дальнейшем программу можно улучшить, путём добавления графического интерфейса, что позволит улучшить навигацию по меню программы, путём добавления возможности использования мыши.

Список используемых источников

1. Язык C++ и программирование на нём: учебное пособие /

В.И. Рейзлин ; Томский политехнический университет. – 3-е изд., перераб. – Томск : Изд-во Томского политехнического университета, 2021. – 208 с.

2. Ассемблер в примерах и задачах: учебное пособие / О.В. Гаркуша, Н.Ю. Добровольская; Министерство науки и высшего образования Российской Федерации, Кубанский государственный университет. − Краснодар: Кубанский гос. ун-т, 2022. − 134 с. − 500 экз.

3. Керниган Б. Ритчи Д. Язык программирования С. 1985 г.

4.MSDN

Приложение А

Листинг программы

Файл calc.cpp

```
#define _CRT_SECURE_NO_WARNINGS
#include <stack>
#include <iostream>
#include <string>
#include <conio.h>
#include <math.h>
#include <windows.h>
#include <stdio.h>
using namespace std;


struct Leksema
{
	char type;
	double value;
};


void decimal_to_binary(int num) {
	if (num < 0) {
		num = num * -1;
		int binary[32];
		int i = 0;
		int b = 0;
		while (num > 0) {
			binary[i] = num % 2;
			num = num / 2;
			i++;
		}
		for (int j = i - 1; j >= 0; j--) {
			if (binary[j] == 1) {
				binary[j] = 0;
				b += 1;
			}
			else if (binary[j] == 0) {
				binary[j] = 1;
				b += 1;
			}
		}
		for (int i = 0; i < b; i++) {
			if (binary[i] != 0) {
				binary[i] = 0;
			}
			else {
				binary[i] = 1;
				break;
			}
		}
		for (int j = i - 1; j >= 0; j--) {
			printf("%d", binary[j]);
		}
		printf("\n");
	}
	else if (num == 0) {
		printf("0\n");
		return;
	}
	else {
		int binary[32];
		int i = 0;
```

```
		while (num > 0) {
			binary[i] = num % 2;
			num = num / 2;
			i++;
		}
		for (int j = i - 1; j >= 0; j--) {
			printf("%d", binary[j]);
		}
		printf("\n");
	}
}
bool Maths(stack <Leksema>& Stack_n, stack <Leksema>& Stack_o, Leksema& item) {
	double a, b, c;
	a = Stack_n.top().value;
	Stack_n.pop();
	switch (Stack_o.top().type) {
	case '+':
		b = Stack_n.top().value;
		Stack_n.pop();
		c = a + b;
		item.type = '0';
		item.value = c;
		Stack_n.push(item);
		Stack_o.pop();
		break;

	case '-':
		b = Stack_n.top().value;
		Stack_n.pop();
		c = b - a;
		item.type = '0';
		item.value = c;
		Stack_n.push(item);
		Stack_o.pop();
		break;

	case '*':
		b = Stack_n.top().value;
		Stack_n.pop();
		c = a * b;
		item.type = '0';
		item.value = c;
		Stack_n.push(item);
		Stack_o.pop();
		break;

	case 's':
		c = sin(a);
		item.type = '0';
		item.value = c;
		Stack_n.push(item);
		Stack_o.pop();
		break;

	case 'c':
		c = cos(a);
		item.type = '0';
		item.value = c;
		Stack_n.push(item);
		Stack_o.pop();
		break;

	case 'd': {
		double dec_number = a;
		decimal_to_binary(dec_number);
		c = dec_number;
		item.type = '0';
		item.value = c;
		Stack_n.push(item);
```

```
			Stack_o.pop();
			break; }
		case 't':
			c = tan(a);
			item.type = '0';
			item.value = c;
			Stack_n.push(item);
			Stack_o.pop();
			break;

		case 'g':
			c = cos(a) / sin(a);
			item.type = '0';
			item.value = c;
			Stack_n.push(item);
			Stack_o.pop();
			break;

		case 'e':
			c = exp(a);
			item.type = '0';
			item.value = c;
			Stack_n.push(item);
			Stack_o.pop();
			break;

		case '^':
			b = Stack_n.top().value;
			Stack_n.pop();
			c = pow(b, a);
			item.type = '0';
			item.value = c;
			Stack_n.push(item);
			Stack_o.pop();
			break;

		case '/':
			b = Stack_n.top().value;
			if (a == 0) {
				cerr << "\nНа 0 делить нельзя!\n";
				return false;
			}
			else {
				Stack_n.pop();
				c = (b / a);
				item.type = '0';
				item.value = c;
				Stack_n.push(item);
				Stack_o.pop();
				break;
			}
		default:
			cerr << "\n Erorr\n";
			return false;
			break;
		}
		return true;
}

int Parara(char Ch) {
		if (Ch == 's' || Ch == 'c' || Ch == 't' || Ch == 'g' || Ch == 'e')
			return 4;
		if (Ch == '^')
			return 3;
		if (Ch == '+' || Ch == '-')
			return 1;
		if (Ch == '*' || Ch == '/')
			return 2;
		else return 0;
```

```
}

void asm_funk() {
	for (int i = 0; i < 5; ++i)
	{
		string s;
		for (int j = 0; j < 50; ++j)
		{
			s.push_back(j < i % 50 ? '#' : ' ');
			printf("    _        _ _        ___      _         _      _
\n");
			printf("   | |_  ___| | |___   / __|__ _| |__ _  _| |__ _| |_ ___
_ _  \n");
			printf("   | ' \\/ -_| | / _ \\ | (__/ _` | / _| || | / _` |  _/
_ | '_| \n");
			printf("   |_||_\\___|_|_\\___/
\\___\\__,_|_\\__|\\_,_|_\\__,_|\\__\\___|_|   \n\n");

		}
		s.push_back('\r');
		Sleep(150);
	}
	printf("Добро пожаловать в калькулятор!\nРукводство пользователя: просто
введите требуемое выражение в строку, все результаты вычислений записываются в
файл\nДоступные тригонометрические функции (в радианах): sin(x); cos(x);
tan(x);ctg(x)\n");
	printf("Длявозведения в степень x впишите ^ x, для извлечения корня степени x
впишите ^ (1 / x)\n");
		printf("Для вычисления дополнительного кода из десятичного числа x
впишите dob(x)\n");
	printf("Для использования экспоненты впишите e, для вычисления экспоненты от
числа x впишите Exp(x)\n");
	printf("Для использования числа Пи впишите p\n");
}

int main() {
	setlocale(LC_ALL, "Rus");
	__asm {
		call asm_funk
	}
	printf(" \n");
	stack <Leksema> Stack_n;
	stack <Leksema> Stack_o;
	Leksema item;
	char Ch;
	double value;
	bool karibo = 1;
	int g = 1;
	while (1) {
		while (52) {
			Ch = cin.peek();
			if (Ch == '\n') {
				break;
			}
			if (Ch == ' ') {
				cin.ignore();
				continue;
			}
			if (Ch == 'p') {
				item.type = '0';
				item.value = acos(-1);
				Stack_n.push(item);
				karibo = 0;
				cin.ignore();
				continue;
			}
			if (Ch == 'e') {
				item.type = '0';
				item.value = exp(1);
```

```
                        Stack_n.push(item);
                        karibo = 0;
                        cin.ignore();
                        continue;
                  }
                  if (Ch == 't' || Ch == 's' || Ch == 'c' || Ch == 'e' || Ch == 'd'
|| Ch == 'E') {
                        char lulu[3];
                        for (short i = 0; i < 3; i++) {
                              Ch = cin.peek();
                              lulu[i] = Ch;
                              cin.ignore();
                        }
                        if (lulu[0] == 's' && lulu[1] == 'i' && lulu[2] == 'n') {
                              item.type = 's';
                              item.value = 0;
                              Stack_o.push(item);

                              continue;
                        }
                        if (lulu[0] == 'd' && lulu[1] == 'o' && lulu[2] == 'b') {
                              item.type = 'd';
                              item.value = 0;
                              Stack_o.push(item);

                              continue;
                        }
                        if (lulu[0] == 'c' && lulu[1] == 'o' && lulu[2] == 's')
                        {
                              item.type = 'c';
                              item.value = 0;
                              Stack_o.push(item);

                              continue;
                        }
                        if (lulu[0] == 't' && lulu[1] == 'a' && lulu[2] == 'n')
                        {
                              item.type = 't';
                              item.value = 0;
                              Stack_o.push(item);

                              continue;
                        }
                        if (lulu[0] == 'c' && lulu[1] == 'a' && lulu[2] == 't')
                        {
                              item.type = 'g';
                              item.value = 0;
                              Stack_o.push(item);
                              continue;
                        }
                        if (lulu[0] == 'E' && lulu[1] == 'x' && lulu[2] == 'p')
                        {
                              item.type = 'e';
                              item.value = 0;
                              Stack_o.push(item);
                              continue;
                        }

                  }

                  if (Ch >= '0' && Ch <= '9' || Ch == '-' && karibo == 1) {
                        cin >> value;
                        item.type = '0';
                        item.value = value;
                        Stack_n.push(item);
                        karibo = 0;
                        continue;
                  }
```

```
                  if (Ch == '+' || Ch == '-' && karibo == 0 || Ch == '*' || Ch ==
'/' || Ch == '^') {
                        if (Stack_o.size() == 0) {
                              item.type = Ch;
                              item.value = 0;
                              Stack_o.push(item);
                              cin.ignore();
                              continue;
                        }
                        if (Stack_o.size() != 0 && Parara(Ch) >
Parara(Stack_o.top().type)) {
                              item.type = Ch;
                              item.value = 0;
                              Stack_o.push(item);
                              cin.ignore();
                              continue;
                        }
                        if (Stack_o.size() != 0 && Parara(Ch) <=
Parara(Stack_o.top().type)) {
                              if (Maths(Stack_n, Stack_o, item) == false) {
                                    system("pause");
                                    return 0;
                              }
                              continue;
                        }
                  }
                  if (Ch == '(') {
                        item.type = Ch;
                        item.value = 0;
                        Stack_o.push(item);
                        cin.ignore();
                        continue;
                  }
                  if (Ch == ')') {
                        while (Stack_o.top().type != '(') {
                              if (Maths(Stack_n, Stack_o, item) == false) {
                                    system("pause");
                                    return 0;
                              }
                              else continue;

                        }
                        Stack_o.pop();
                        cin.ignore();
                        continue;
                  }
                  else {
                        printf("error");
                        return 0;
                  }
            }
            while (Stack_o.size() != 0) {
                  if (Maths(Stack_n, Stack_o, item) == false) {
                        system("pause");
                        return 0;
                  }
                  else
                        continue;
            }
            printf("Ответ: %lf \n", Stack_n.top().value);
            cin.ignore();
            FILE* fptr;
            fptr = fopen("fileName.txt", "a+");
            fprintf(fptr, "Ответ: %lf", Stack_n.top().value);
            fprintf(fptr, "\n");
            fclose(fptr);
            continue;
      }
}
```

Приложение В

Результат вывода программы

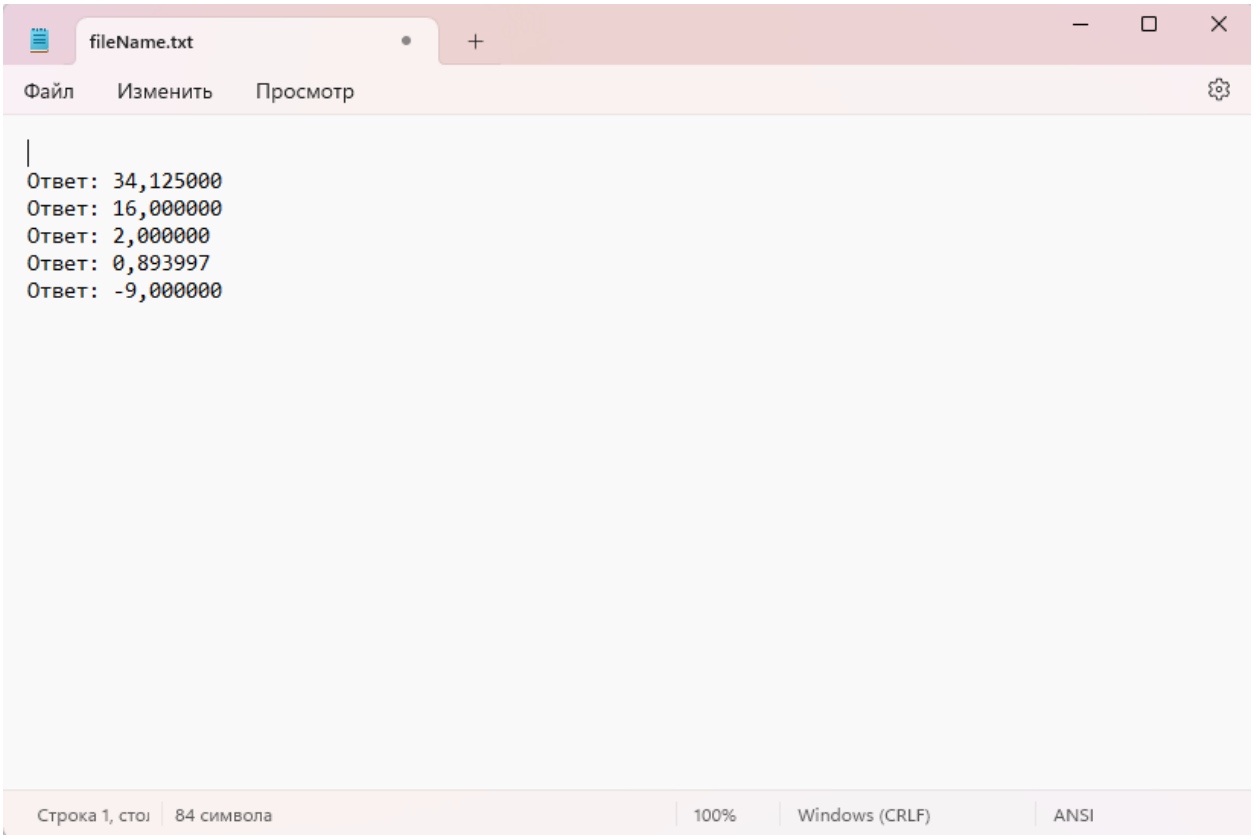


Рисунок 15 – вывод программы