

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

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

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

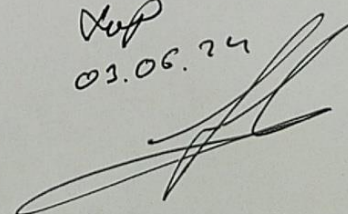
Выполнил:

студент группы 23BVB4

Святов И. Ю.

Принял:

Голотенков Н. О.

Н.О.
03.06.24


Пенза, 2024

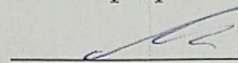
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Вычислительной техники

Кафедра “Вычислительная техника”

“УТВЕРЖДАЮ”

Зав. кафедрой ВТ



М.А. Митрохин

«___» _____ 2024 г.

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Святову Ивану Юрьевичу 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 функции ($\sin$, $\cos$, tg , ctg), переводить десятичные числа в дополнительный код, также результаты технических вычислений программа должна сохранять 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 “ 9 ” февраля 2024 г.

Дата защиты проекта “ 05 ” 06 . 2024 г.

Руководитель Голотенков Николай Олегович

Задание получил “9” февраля 2024 г.

Студент Святов Иван Юрьевич

Содержание

Введение.....	5
Постановка задачи.....	6
Выбор решения.....	6
Описание разработки программы.....	7
Отладка и тестирование.....	8
Описание программы.....	12
Руководство пользователя.....	20
Заключение.....	24
Список используемой литературы.....	25
Приложение А Листинг программы.....	26
Приложение В Результаты работы программы.....	34
Приложение С Результат работы программы.....	36

Введение

В настоящее время повсеместное использование калькуляторов существенно облегчает работу человека в самых различных сферах, ведь практически в каждой отрасли деятельности нужно проводить точные подсчеты. Калькулятор во многом упрощает ручные подсчеты, то есть позволяет проводить математические операции с максимальной точностью, избавившись от человеческих ошибок в расчетах.

Инженерный калькулятор – это устройство или программа, которое позволяет выполнять различные математические операции, такие как сложение, вычитание, умножение и деление, возведение в степень, извлечение квадратного корня и так далее. В наше время калькуляторы являются неотъемлемой частью повседневной жизни, используя их для решения различных задач как в учебных заведениях, так и в повседневной деятельности.

Первый калькулятор массового выпуска был изобретен в 1961 году и назывался ANITA MK VIII. Изобретение выпускалось в Англии и представляло собой устройство, работающее на газоразрядных лампах. Это устройство было достаточно громоздким по современным меркам, оно оснащалось клавиатурой для ввода числа, а также дополнительной 10-ти клавишной консолью для задания множителя. В 1965-ом году калькуляторы Wang впервые научились считать логарифмы, а уже через четыре года в США появился и первый настольный программируемый калькулятор. А в 1970-ые годы мир калькуляторов становится более совершенным и разнообразным - появляются новые настольные и карманные машины, а также профессиональные инженерные калькуляторы, позволяющие производить сложнейшие расчеты.

Таким образом, в рамках данного курсового проекта будет разрабатываться программа приложения «Инженерный калькулятор».

Постановка задачи

Необходимо разработать программу «Инженерный калькулятор». Калькулятор должен выполнять сложение, вычитание, умножение, деление, возведение в степень, извлечение квадратного корня, а также вычисление тригонометрических функций ($\sin$, $\cos$, tg , ctg) и переводить десятичные числа в дополнительный код. Программа должна иметь интуитивно понятный интерфейс для пользователя.

Многомодульность программы. Это организация программы как совокупности небольших независимых блоков, называемых модулями, то есть программа должна быть поделена на логические модули. Это упростит поиск ошибок при отладке и тестировании, а также позволит легко расширять функционал программы.

Использование сложных типов данных – массивов, структур, файлов. Необходимо изучить и использовать в программе сложные типы данных.

Режим работы видеосистемы – текстовый/графический. Перед разработкой программы необходимо выбрать тип интерфейса, а также элементы управления программой.

Устройство ввода-вывода – клавиатура и мышь. Необходимо анализировать и различать события, возникающие от работы с клавиатурой и мышью. Необходимо идентифицировать и выполнять те или иные действия в зависимости от действий пользователя.

Пользовательский интерфейс должен быть построен на основе меню и панели инструментов. Это необходимо того, чтобы интерфейс, который будет использовать пользователь был интуитивно понятен.

Наличие заставки. Заставка необходима для того, чтобы пользователь, который запускает программу смог понять, что представляет из себя, и как работает программа.

Выбор решения

При запуске программы сначала идет заставка, а после открывается главный экран, на котором изображено меню, состоящее из трех пунктов.

Навигация по данному меню осуществляется вводом цифр с клавиатуры, от 0 до 2.

1. Вычисление выражения. Пользователь вводит выражение, которое нужно вычислить, расставляя символы «(» и «)», в тех местах, где нужен приоритет операций. Также для данного пункта представлены все возможные действия, а именно: «+» – операция сложения, «-» – операция вычитания, «*» – операция умножения, «/» – операция деления, «^» – операция возведения в степень, «\$» – операция вычисления корня, «s» – операция вычисления синуса, «c» – операция вычисления косинуса, «t» – операция вычисления тангенса, «g» – операция вычисления котангенса.
2. Вычисление дополнительного кода. Пользователь вводит сначала число, которое нужно преобразовать, а потом количество бит, которое нужно вывести. После программа преобразует введенное число в дополнительный код.
3. Завершение программы. Выход из программы.

Описание разработки программы

Для написания курсового проекта был использован язык программирования СИ, с применением ассемблера. Этот язык является одним из самых распространённых языков программирования. Целью языка было облегчение написания больших программ с минимизацией ошибок по сравнению с ассемблером, следуя принципам процедурного программирования, но избегая всего, что может привести к дополнительным накладным расходам, специфичным для языков высокого уровня. К тому же, несмотря на свою низкоуровневую природу, язык ориентирован на переносимость. Программы, соответствующие стандарту языка, могут компилироваться под различные архитектуры компьютеров.

Разработанная программа состоит из четырех модулей, которые реализовывались в следующем порядке:

1. `kursovoi.cpp` – основной файл программы.

2. Calc.cpp – файл для вычисления введенного математического выражения, который обрабатывает все доступные в основной программе знаки.
3. AdditionalCode.cpp – файл для перевода десятичного числа в дополнительный код.
4. ScreenSaver.cpp – файл, в котором хранится заставка программы.

В данном случае по порядку реализации моделей программы, можно сделать вывод, что использовался нисходящий подход реализации. Этот метод основан на том, что сначала проектируются основные компоненты программы, а затем уже дорабатываются ее мелкие детали.

Отладка и тестирование

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2019. Программа обладает всеми средствами необходимыми при разработке и отладке программы, а именно: возможность поставить точки останова, пошаговое выполнение программы, а также отслеживание и анализ содержимого переменных.

В документации по отладке для Visual Studio под отладкой обычно подразумевается узкоспециализированное средство разработки, которое присоединяется к работающему приложению и позволяет проверять код.

Точка останова – это один из важнейших инструментов разработчика для выполнения отладки, который устанавливается везде, где нужно приостановить выполнение отладчика и проанализировать переменные на данном этапе программы. Точку останова можно поставить на любой строчке исполняемого кода. Для того, чтобы поставить точку, необходимо нажать мышкой в крайнем левом поле, около исполняемого кода, после чего в этом месте появится красная точка, которая и является точкой останова. Можно также выбрать строку и нажать клавишу F9, выбрать команду «Отладка» после чего выбрать «Переключить точку останова».

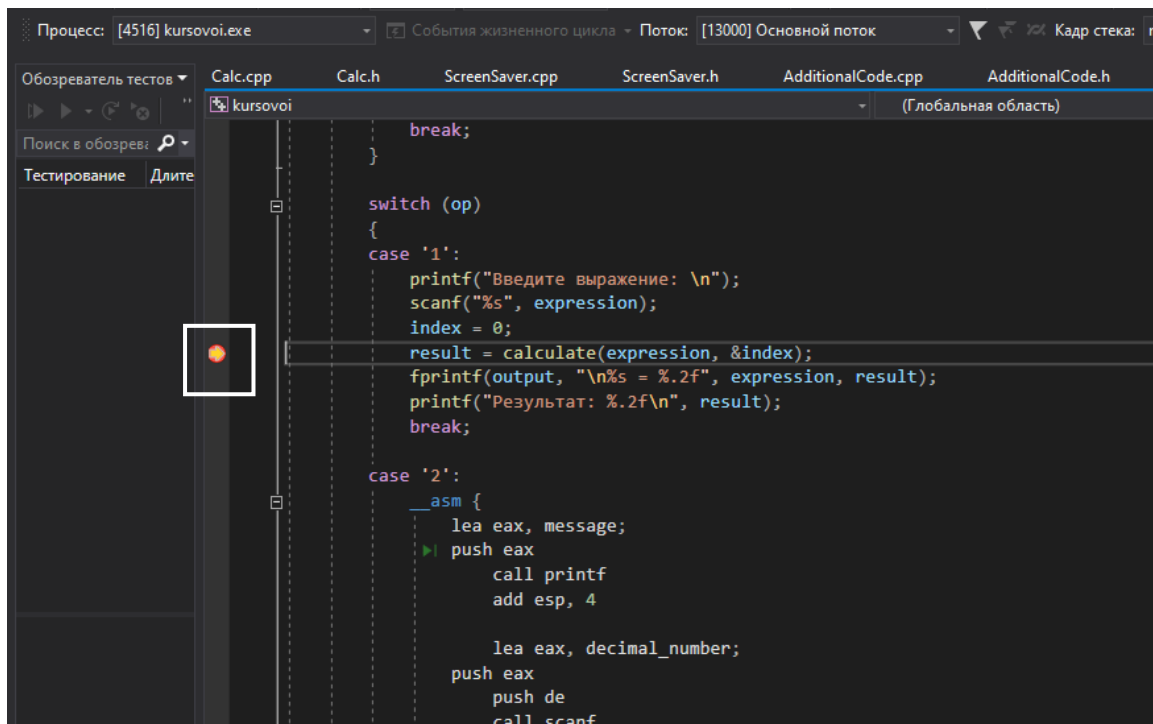


Рисунок 1 – Отладка программы при помощи точки останова

Пошаговое выполнение программы – это еще одно средство отладки программы, с помощью которого можно переходить по коду для проверки состояния приложения и просмотра последовательности выполнения. Данное средство отладки можно запустить при помощи клавиши «F10». Чтобы переключиться на следующую строку кода необходимо также нажимать клавишу «F10». Пошаговое выполнение программы может быть: с заходом в функцию и без захода в функцию. В первом случае при выполнении кода программа перейдет в функцию и можно отслеживать переменный в самой функции, по окончании функции программа перейдет к месту вызова и продолжит выполнять код. В данном случае, чтобы зайти в функцию необходимо нажать кнопку на панели инструментов «Шаг с заходом» или клавишу «F11» на клавиатуре, представленный на рисунке 2. Во втором случае программа выполнит код без захода в функцию. Чтобы выполнять код и не заходить в функцию необходимо нажать кнопку на панели инструментов «Шаг с обходом» или клавишу «F10» на клавиатуре, представленный на рисунке 3.

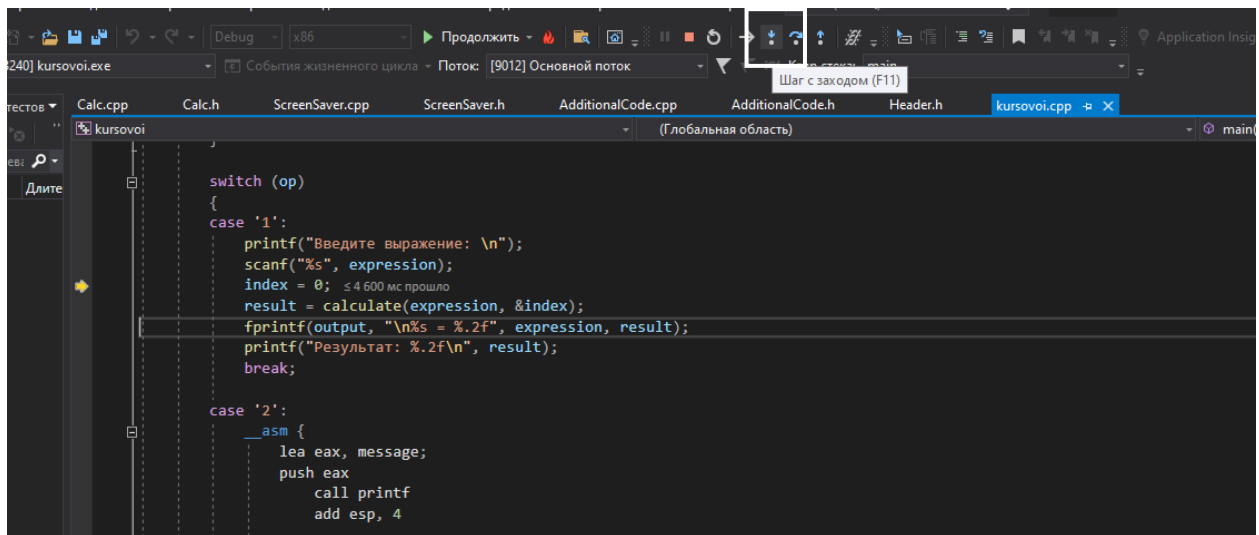


Рисунок 2 – Пошаговое выполнение программы с заходом в функцию

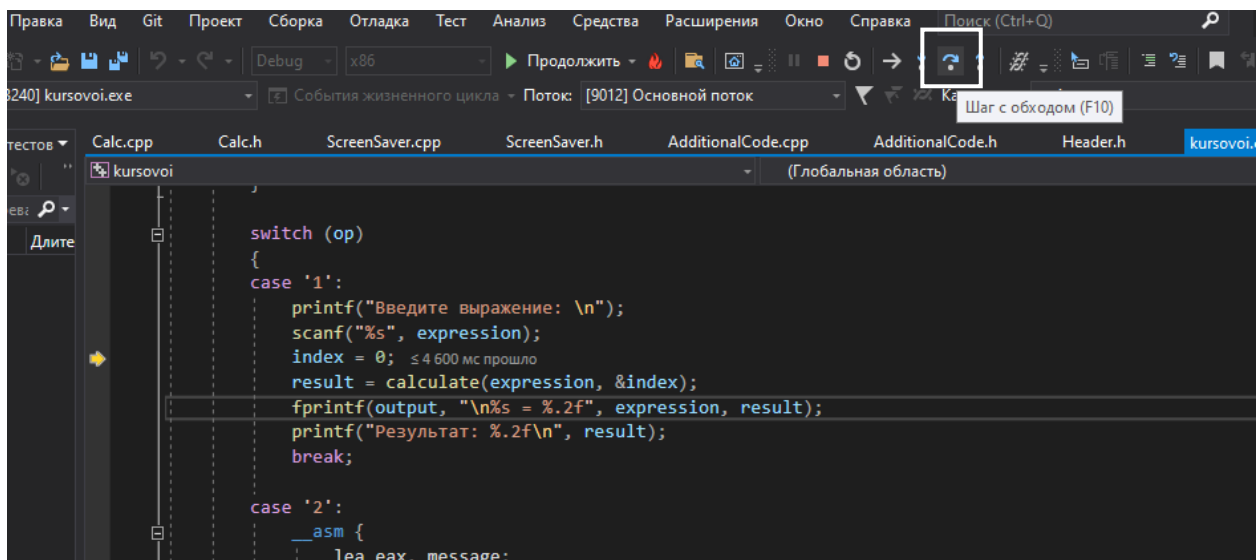


Рисунок 3 – Пошаговое выполнение программы с обходом функции

Чтобы открыть окно «Контрольных значений», при помощи которого можно отслеживать изменения переменных по ходу всей программы, необходимо поставить точку останова и запустить отладку, или запустить пошаговое выполнение программы, затем открыть окно «Контрольные значения», выбрав «Отладка» – «Окна» – «Контрольные значения» – «Контрольные значения 1». Весь путь представлен на рисунке 4.

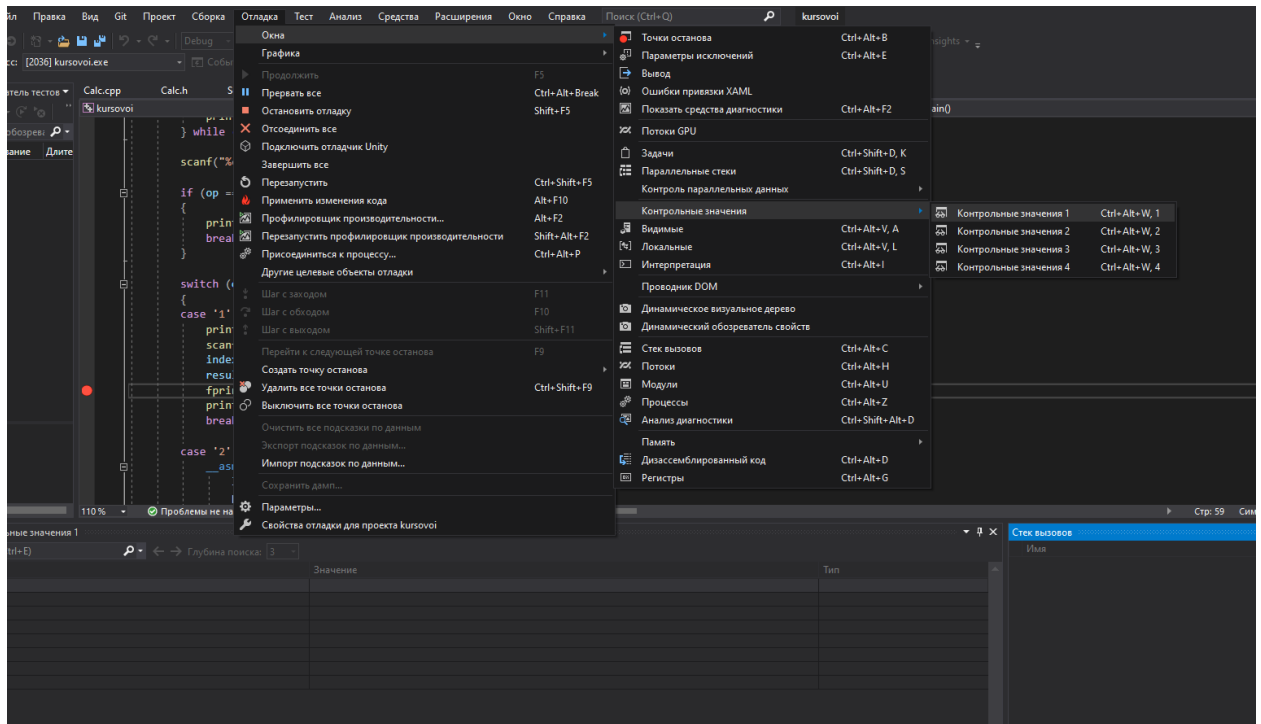


Рисунок 4 – Открытие окна контрольных значений

После открытия окна необходимо выбрать пустую строку и задать переменную, которую надо отслеживать. В данном окне можно увидеть имя, значение, а также тип отслеживаемой переменной. Пример представлен на рисунке 5.

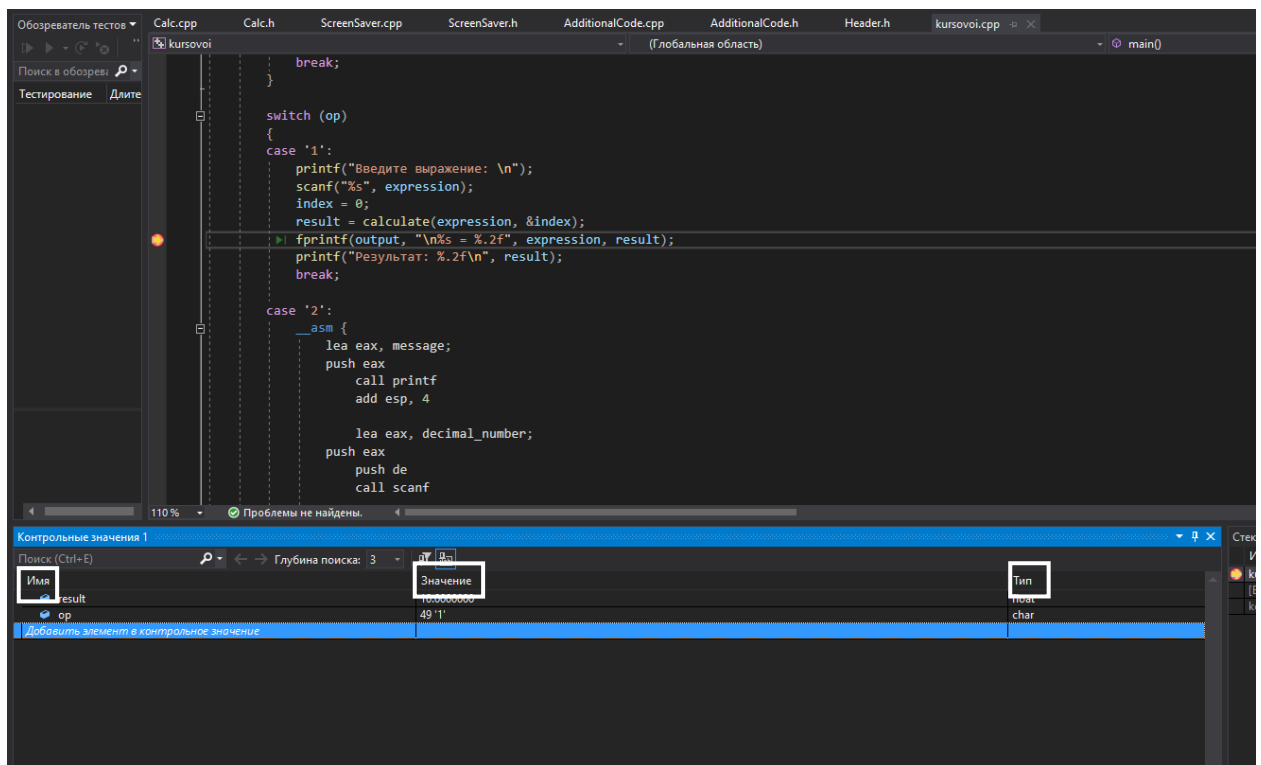


Рисунок 5 – Окно контрольных значений

Тестирование проводилось во время разработки и также после завершения.

В ходе разработки было выявлено и исправлено большое количество проблем, связанных с неправильностью подсчетов, сохранения результатов в файл, неправильностью перевода десятичных чисел в дополнительный код.

Описание программы

Приложение kursovoi.exe является основным модулем программы. При запуске сначала на экран выводится заставка, после чего, по истечению пяти секунд появляется главное окно, в котором пользователю необходимо выбрать пункт меню для дальнейшего использования калькулятора. Выход из программы осуществляется выбором соответствующего пункта меню. Подробное описание действий программы, при выборе конкретных пунктов меню представлено в таблице ниже.

Клавиши, вызывающие событие	Действие пользователя	Действие программы
1, Enter	Выбран пункт меню «Вычисление выражения».	Запускается диалог, в котором пользователю необходимо ввести математическое или тригонометрическое выражение, после чего вычисляется и выводится на экран результат выполнения.
2, Enter	Выбран пункт меню «Вычисления дополнительного кода»	Запускается диалог, в котором пользователю необходимо ввести десятичное число, а также количество бит, после чего вычисляется и выводится на экран результат выполнения.
3, Enter	Выбран пункт меню «Завершение программы».	Выполняется закрытие программы.

Таблица 1 – Описание действий программы

Ниже представлены схемы программы для функций main, calculate, DopCode.



Рисунок 6 – Схема функции main (Часть 1)



Рисунок 7 – Схема функции `main` (Часть 2)

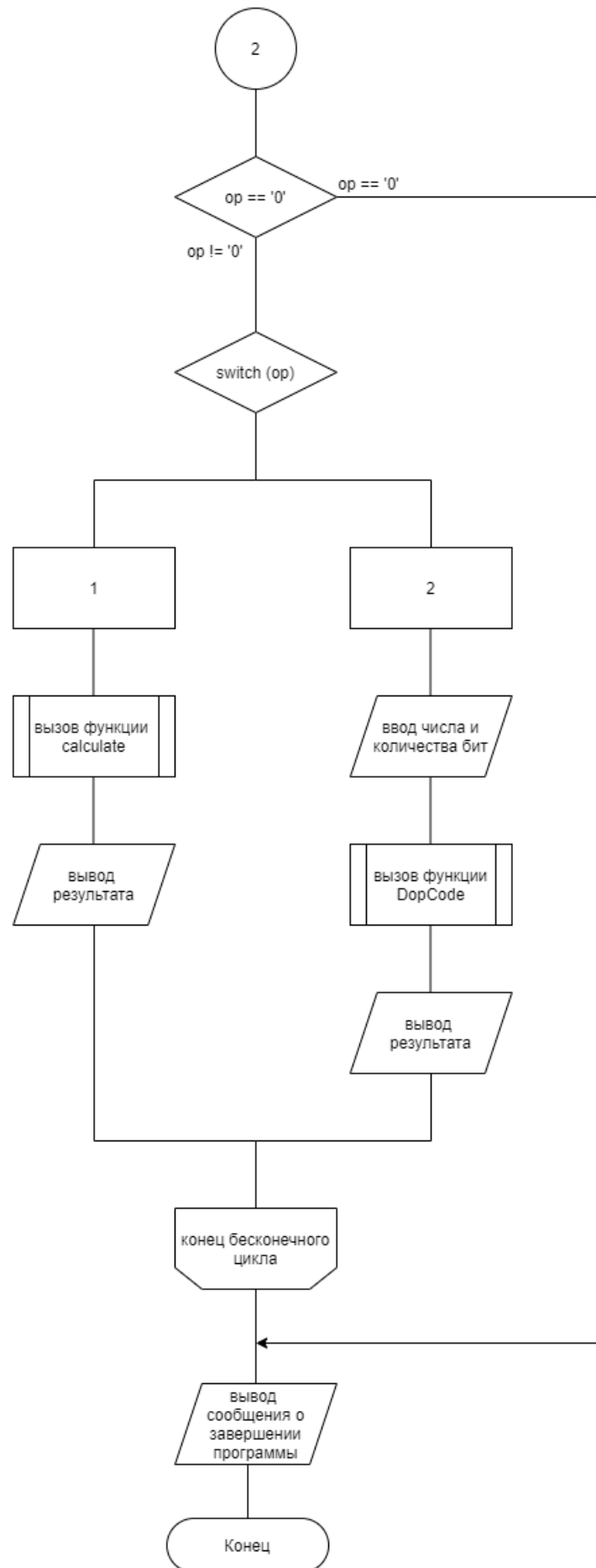


Рисунок 8 – Схема функции main (Часть 3)

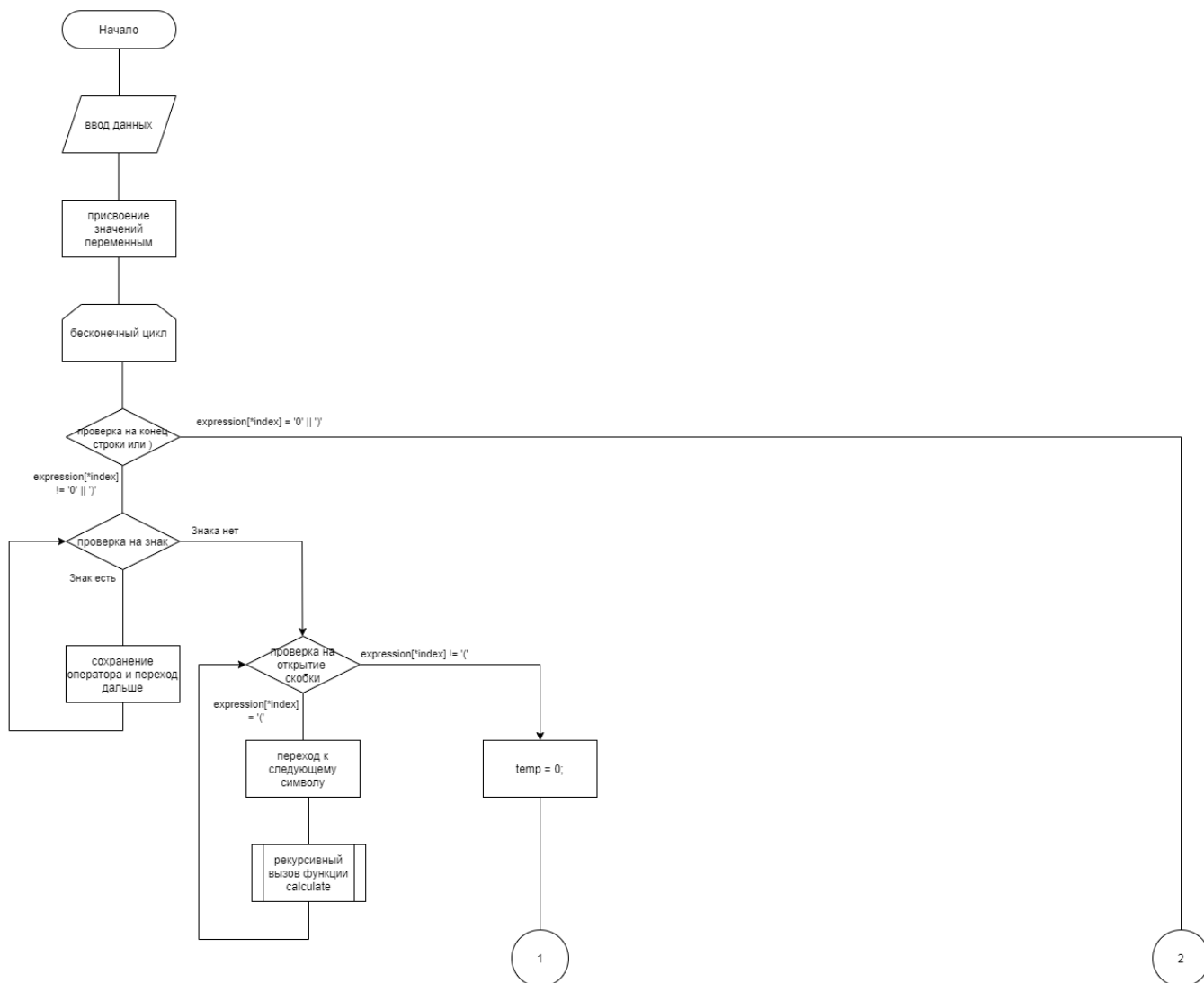


Рисунок 9 – Схема функции calculate (Часть 1)

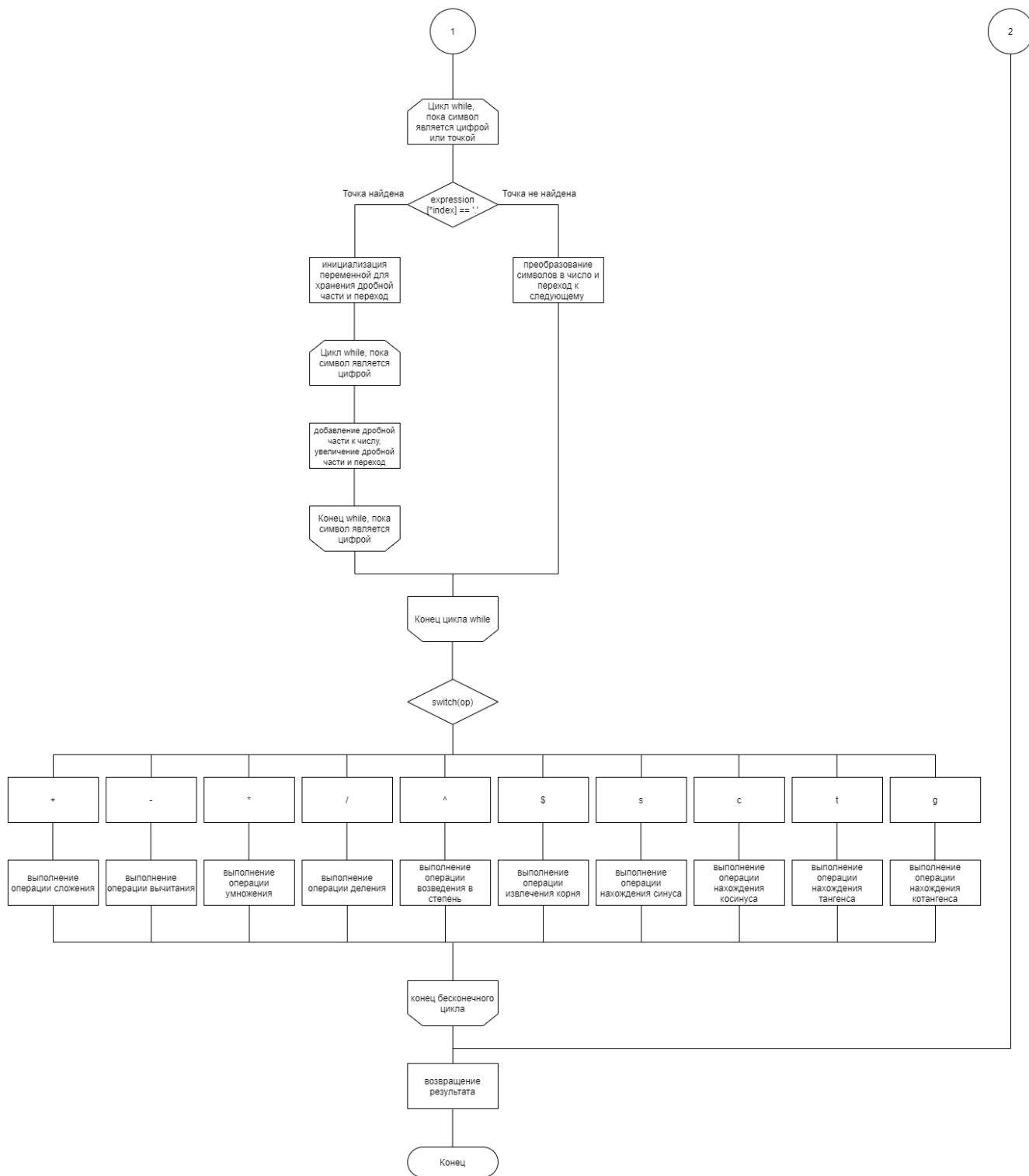


Рисунок 10 – Схема функции calculate (Часть 2)

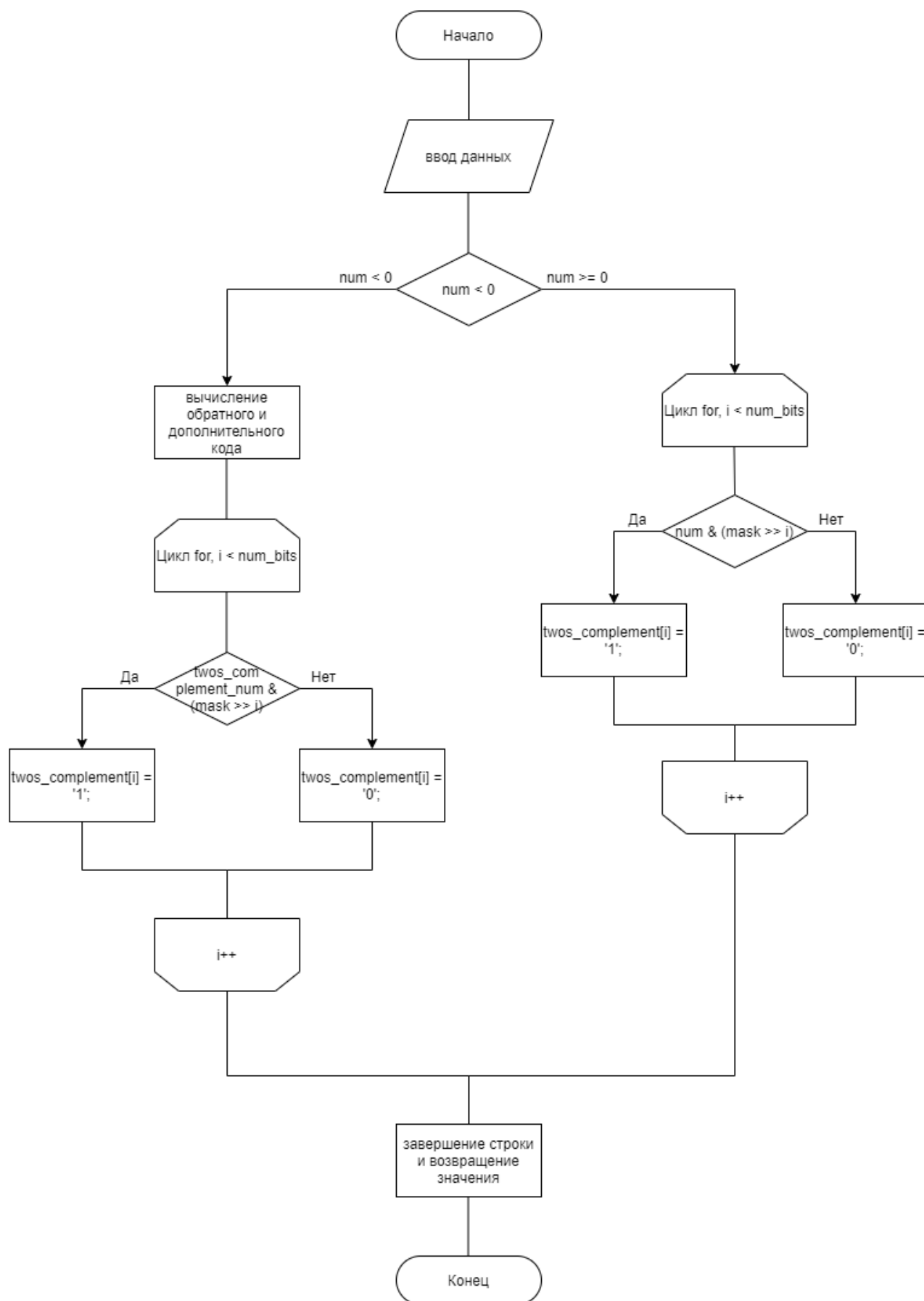


Рисунок 11 – Схема функции DorCode

Также для данного курсового проекта была спроектирована иерархическая

схема программы, которая представлена на рисунке 12.

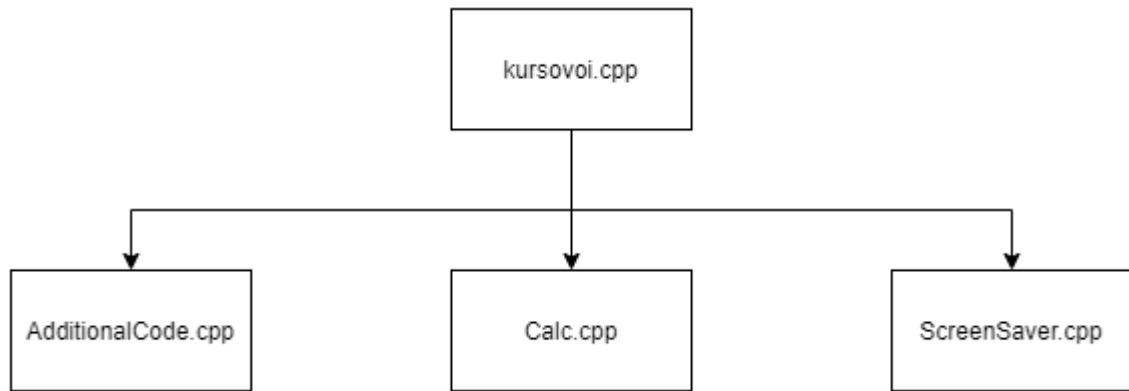


Рисунок 12 – Иерархическая схема программы

Меню реализовано в файле `kursovoi.cpp` и состоит из пунктов: вычисление выражения, вычисление дополнительного кода, а также завершение программы.

Главной задачей данного меню является легкая и понятная навигация нового пользователя в данной программе. Навигация реагирует на нажатие клавиш на клавиатуре от 0 до 2.

Меню располагается в самом верху окна, чтобы пользователь всегда мог видеть всевозможные действия.

В данном курсовом проекте применяется ассемблерная вставка. В качестве фрагмента программы, в котором код, написанный на C, будет заменён на ассемблерный, был выбран фрагмент с выводом сообщений и вводом чисел для функции вычисления дополнительного кода.

Компилятор, входящий в состав среды разработки Visual Studio 2019, позволяет сделать ассемблерную вставку прямо в коде, написанном на языке программирования C. Чтобы это реализовать необходимо перед ассемблерной вставкой написать ключевое слово «`__asm`», а само тело поместить в фигурные скобки.

Для реализации данного фрагмента кода были использованы следующие ассемблерные инструкции:

- инструкция `lea` вычисляет эффективный адрес источника и помещает его в приёмник;
- инструкция `push` помещает какое-либо значение в стек программы;
- инструкция `call` передаёт управление процедуре, а также помещает в

стек 64-битный адрес инструкции, которая идет сразу после вызова;

- инструкция add выполняет сложение двух чисел.

Данный фрагмент выполняет вывод информационных сообщений программы, а также пользовательский ввод числа, представленного в десятичной системе счисления и ввод количества бит числа.

Руководство пользователя

Программа kursovoi.exe предназначена для проведения расчетов. В программе присутствует интуитивно понятный интерфейс и поддерживаются такие операции, как сложение, вычитание, умножение, деление, возведение в степень, вычисление квадратного корня, вычисление тригонометрических функций, а именно $\sin$, $\cos$, tg , ctg , а также перевод десятичных чисел в дополнительный код.

Для вычисления математических или тригонометрических выражений необходимо выбрать пункт меню «вычисление выражения», находящееся под цифрой 1.

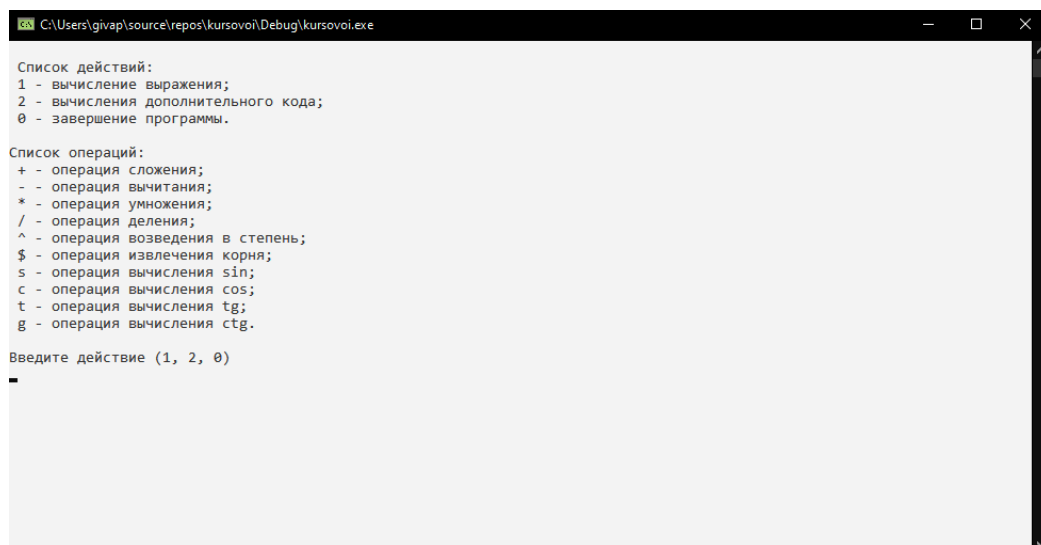


Рисунок 13 – Окно выбора действия

Далее необходимо ввести математическое или тригонометрическое выражение. Для ввода выражений под списком действий введены все допустимые операторы, которые может обрабатывать программа. После ввода и нажатия на клавишу «Enter», программа выполнит расчеты и выведет конечный результат.

```
C:\Users\givap\source\repos\kursovoi\Debug\kursovoi.exe

Список действий:
1 - вычисление выражения;
2 - вычисления дополнительного кода;
0 - завершение программы.

Список операций:
+ - операция сложения;
- - операция вычитания;
* - операция умножения;
/ - операция деления;
^ - операция возведения в степень;
$ - операция извлечения корня;
s - операция вычисления sin;
c - операция вычисления cos;
t - операция вычисления tg;
g - операция вычисления ctg.

Введите действие (1, 2, 0)
1
Введите выражение:
5+7-(6*3)
```

Рисунок 14 – Ввод выражения

```
C:\Users\givap\source\repos\kursovoi\Debug\kursovoi.exe

Список действий:
1 - вычисление выражения;
2 - вычисления дополнительного кода;
0 - завершение программы.

Список операций:
+ - операция сложения;
- - операция вычитания;
* - операция умножения;
/ - операция деления;
^ - операция возведения в степень;
$ - операция извлечения корня;
s - операция вычисления sin;
c - операция вычисления cos;
t - операция вычисления tg;
g - операция вычисления ctg.

Введите действие (1, 2, 0)
1
Введите выражение:
5+7-(6*3)
Результат: -6.00
```

Рисунок 15 – Результат выражения

Для того, чтобы вычислить дополнительный код любого десятичного числа необходимо выбрать пункт меню «вычисления дополнительного кода», находящееся под цифрой 2. После выбора соответствующей цифры, необходимо будет ввести сначала само десятичное число, которое необходимо преобразовать.

```
C:\Users\givap\source\repos\kursovoi\Debug\kursovoi.exe

Список действий:
1 - вычисление выражения;
2 - вычисления дополнительного кода;
0 - завершение программы.

Список операций:
+ - операция сложения;
- - операция вычитания;
* - операция умножения;
/ - операция деления;
^ - операция возведения в степень;
$ - операция извлечения корня;
s - операция вычисления sin;
c - операция вычисления cos;
t - операция вычисления tg;
g - операция вычисления ctg.

Введите действие (1, 2, 0)
2
Введите десятичное число:
-10
```

Рисунок 16 – Ввод десятичного числа

После ввода числа потребуется ввести также количество бит для вывода.

```
C:\Users\givap\source\repos\kursovoi\Debug\kursovoi.exe

Список действий:
1 - вычисление выражения;
2 - вычисления дополнительного кода;
0 - завершение программы.

Список операций:
+ - операция сложения;
- - операция вычитания;
* - операция умножения;
/ - операция деления;
^ - операция возведения в степень;
$ - операция извлечения корня;
s - операция вычисления sin;
c - операция вычисления cos;
t - операция вычисления tg;
g - операция вычисления ctg.

Введите действие (1, 2, 0)
2
Введите десятичное число:
-10
Введите количество бит числа:
8
```

Рисунок 17 – Ввод количества бит

В конечном итоге, после ввода всех необходимых переменных, программа вычислит дополнительный код введенного десятичного числа и выведет результат.

```
C:\Users\givap\source\repos\kursovoi\Debug\kursovoi.exe

Список действий:
1 - вычисление выражения;
2 - вычисления дополнительного кода;
0 - завершение программы.

Список операций:
+ - операция сложения;
- - операция вычитания;
* - операция умножения;
/ - операция деления;
^ - операция возведения в степень;
$ - операция извлечения корня;
s - операция вычисления sin;
c - операция вычисления cos;
t - операция вычисления tg;
g - операция вычисления ctg.

Введите действие (1, 2, 0)
2
Введите десятичное число:
-10
Введите количество бит числа:
8
Десятичное число: -10
Дополнительный код: 11110110
```

Рисунок 18 – Результат вычисления дополнительного кода

Для того, чтобы завершить программу необходимо выбрать пункт меню «завершение программы», находящееся под цифрой 0. После выбора данного пункта меню, в программе больше нельзя будет выбрать ни одно действие.

Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Была освоена работа с файлами, изучены функции работы с консолью и оперативной памятью. Были освоены приемы создания псевдографического интерфейса. Получены навыки реализации меню. Также были получены основные навыки программирования, а также отладки и тестирования в среде Visual Studio 2019 на языках Си и Assembler.

В рамках курсовой работы была написана программа «Инженерный калькулятор», с помощью которой можно вычислять различные математические выражения. Программа быстро производит вычисления и не требует большого количества системных ресурсов.

В дальнейшем данную программу можно улучшить, добавив графический интерфейс, чтобы было удобнее пользователю, а также, добавив ещё больше математических функций, таких как: логарифмы, интегралы, пределы, дифференциалы и другие, так как это все расширит функционал программы.

Список используемой литературы

1. Керниган Б. Ритчи Д. Язык программирования С. 1985 г.
2. Внедрение кода ассемблера в код С: [сайт] – URL: <https://metanit.com/assembler/arm64/6.3.php>. (дата обращения: 10.05.2024).
3. Многомодульное программирование на языке С: [сайт] – URL: https://foxford.ru/wiki/informatika/modulnoe-programmirovaniye-na-s?utm_referrer=https%3A%2F%2Fyandex.ru%2F. (дата обращения: 28.04.2024).
4. Работа с файлами в языке С: [сайт] – URL: <https://prog-cpp.ru/c-files/>. (дата обращения: 02.05.2024).
5. Организация заставки на языке С: [сайт] – URL: <https://qna.habr.com/q/292446>. (дата обращения: 25.04.2024).
6. Прямой, обратный и дополнительный коды: [сайт] – URL: <https://microkontroller.ru/programmirovaniye-mikrokontrollerov-avr/pryamoy-obratnyiy-dopolnitelnyiy-kod-dvoichnogo-chisla/>. (дата обращения: 17.04.2024).

Приложения А
Листинги программы
Файл «kursovoi.cpp»

```
#include "Header.h"

void main() {
    system("chcp 1251");

    zastavka();

    char expression[100];
    char op;
    float result = 0;
    int decimal_number, num_bits;
    int sum = 0;
    int index;
    char message[] = "Введите десятичное число: \n";
    char dec[] = "%d";
    char* de = dec;
    char message2[] = "Введите количество бит числа: \n";
    char bits[] = "%d";
    char* bit = bits;
    FILE* output;
    do {
        output = fopen("history.txt", "w");

        if (output != NULL)
            break;

        printf("Не удалось открыть файл! Повторите
ввод\n\n");
    } while (true);
    fprintf(output, "История операций:");
```

```

        printf("\n Список действий:\n 1 - вычисление выражения;\n
2 - вычисления дополнительного кода;\n 0 - завершение программы.
\n\nСписок операций:\n + - операция сложения;\n - - операция
вычитания;\n * - операция умножения;\n / - операция деления;\n ^ -
операция возведения в степень;\n $ - операция извлечения корня;\n
s - операция вычисления sin;\n c - операция вычисления cos;\n t -
операция вычисления tg;\n g - операция вычисления ctg.\n");
        printf("\nВведите действие (1, 2, 0)\n");

        while (true)
        {
            do {
                output = freopen("history.txt", "a", output);
                if (output != NULL)
                    break;
                printf("Не удалось открыть файл! Повторите
ввод\n\n");
            } while (true);

            scanf("%c", &op);

            if (op == '0')
            {
                printf("Завершение программы...");
                break;
            }

            switch (op)
            {
            case '1':
                printf("Введите выражение: \n");
                scanf("%s", expression);
                index = 0;
                result = calculate(expression, &index);
                fprintf(output, "\n%s = %.2f", expression,
result);

```

```

        printf("Результат: %.2f\n", result);
        break;

    case '2':
        __asm {
            lea eax, message;
            push eax
            call printf
            add esp, 4

            lea eax, decimal_number;
            push eax
            push de
            call scanf

            lea eax, message2;
            push eax
            call printf
            add esp, 4

            lea eax, num_bits
            push eax
            push bit
            call scanf
        }
        printf("Десятичное число: %d\n",
decimal_number);
        printf("Дополнительный код: %s\n",
DopCode(decimal_number, num_bits));
        fprintf(output, "\nДополнительный код числа %d
= %s", decimal_number, DopCode(decimal_number, num_bits));
        break;

        fclose(output);

    }

```

```
    }  
}
```

Файл «Calc.cpp»

```
#include "Header.h"  
  
float calculate(char* expression, int* index) {  
    float result = 0;  
    float temp = 0;  
    float angle = 0;  
    char op = '+';  
  
    while (1) {  
        if (expression[*index] == '\\0' || expression[*index]  
== ')') { // проверка на конец строки или закрывающую скобку  
            break; // выход из цикла  
        }  
  
        if (expression[*index] == '+' || expression[*index]  
== '-' || expression[*index] == '*' || expression[*index] == '/'  
|| expression[*index] == '^' || expression[*index] == '$' ||  
expression[*index] == 's' || expression[*index] == 'c' ||  
expression[*index] == 't' || expression[*index] == 'g') {  
            op = expression[*index]; // сохранение  
оператора  
  
            (*index)++;  
        }  
        else {  
            if (expression[*index] == '(') { // если  
найдена (  
  
                (*index)++;  
                temp = calculate(expression, index); //  
вызов рекурсивно функции  
            }  
            else {  
                temp = 0;  
            }  
        }  
    }  
}
```

```

        while ((expression[*index] >= '0' &&
expression[*index] <= '9') || expression[*index] == '.') { // пока
символ является цифрой или точкой
            if (expression[*index] == '.') { //
если найдена .
                (*index)++;
                float fraction = 0.1; //
инициализация переменную для хранения дробной части
                while (expression[*index] >= '0'
&& expression[*index] <= '9') { // пока символ является цифрой
                    temp += (expression[*index]
- '0') * fraction; // добавление дробной части к числу
                    fraction *= 0.1; //
увеличиваем дробную часть
                    (*index)++;
                }
            }
            else {
                temp = temp * 10 +
(expression[*index] - '0'); // преобразование символов в число
                (*index)++; // след элемент
            }
        }
    }

    switch (op) {
    case '+':
        result += temp;
        break;
    case '-':
        result -= temp;
        break;
    case '*':
        result *= temp;
        break;
    case '/':

```

```

        result /= temp;
        break;
    case '^':
        result = pow(result, temp);
        break;
    case '$':
        result = sqrt(temp);
        break;
    case 's':
        angle = temp * M_PI / 180;
        result = sin(angle);
        break;
    case 'c':
        angle = temp * M_PI / 180;
        result = cos(angle);
        break;
    case 't':
        angle = temp * M_PI / 180;
        result = tan(angle);
        break;
    case 'g':
        angle = temp * M_PI / 180;
        result = 1 / tan(angle);
        break;
    }
}

return result;
}

```

Файл «AdditionalCode.cpp»

```

#include "Header.h"

char* DopCode(int num, int num_bits) {
    static char twos_complement[32]; // массив для хранения
двоичного числа
    int mask = 1 << (num_bits - 1); // создание битовой маски

```

```

        if (num < 0) { // яв-ся ли число отрицательным
            int positive_num = -num; // вычисление модуля
отрицательного числа

            int ones_complement = positive_num ^ ((1 <<
num_bits) - 1); // обратный код

            int twos_complement_num = ones_complement + 1; //
доп код

            for (int i = 0; i < num_bits; i++) { // цикл для
обработки каждого бита числа

                if (twos_complement_num & (mask >> i)) { //
проверяем установлен ли i-й бит

                    twos_complement[i] = '1'; // установка
бита в '1'

                }
                else {
                    twos_complement[i] = '0';// установка бита
в '0'

                }
            }
        }
        else {
            for (int i = 0; i < num_bits; i++) {
                if (num & (mask >> i)) { // цикл для обработки
каждого бита числа

                    twos_complement[i] = '1';

                }
                else {
                    twos_complement[i] = '0';

                }
            }
        }
        twos_complement[num_bits] = '\\0';

        return twos_complement;

```

```
}
```

Файл «ScreenSaver.cpp»

```
#include "Header.h"
```

```
int zastavka() {
    printf("*****\n");
    printf("* Добро пожаловать *\n");
    printf("* в приложение *\n");
    printf("* КАЛЬКУЛЯТОР *\n");
    printf("* * * *\n");
    printf("* ----- *\n");
    printf("* | | * *\n");
    printf("* | | * *\n");
    printf("* ----- *\n");
    printf("* ----- *\n");
    printf("* | + | | - | | * | | | *\n");
    printf("* ----- *\n");
    printf("* ----- *\n");
    printf("* | 1 | | 2 | | 3 | | / | *\n");
    printf("* ----- *\n");
    printf("* ----- *\n");
    printf("* | 4 | | 5 | | 6 | | ^ | *\n");
    printf("* ----- *\n");
    printf("* ----- *\n");
    printf("* | 7 | | 8 | | 9 | | = | *\n");
    printf("* ----- *\n");
    printf("* * * *\n");
    printf("*****\n");
    // Таймер
    time_t start_time = time(NULL);
    while (time(NULL) < start_time + 5) {}
    // Очищаем экран
    system("cls");
    return 0;
}
```

Приложение В

Результаты работы программы

```
C:\Users\givap\source\repos\kursovoi\Debug\kursovoi.exe
Текущая кодовая страница: 1251
*****
* Добро пожаловать *
* в приложение *
* КАЛЬКУЛЯТОР *
* *
* *
* ----- *
* | | | | | *
* | | | | | *
* ----- *
* | + | | - | | * | *
* ----- *
* | 1 | | 2 | | 3 | | / | *
* ----- *
* | 4 | | 5 | | 6 | | ^ | *
* ----- *
* | 7 | | 8 | | 9 | | = | *
* ----- *
* * *
*****
```

Приложение В.1 – Информационная заставка

```
C:\Users\givap\source\repos\kursovoi\Debug\kursovoi.exe

Список действий:
1 - вычисление выражения;
2 - вычисления дополнительного кода;
0 - завершение программы.

Список операций:
+ - операция сложения;
- - операция вычитания;
* - операция умножения;
/ - операция деления;
^ - операция возведения в степень;
$ - операция извлечения корня;
s - операция вычисления sin;
c - операция вычисления cos;
t - операция вычисления tg;
g - операция вычисления ctg.

Введите действие (1, 2, 0)
1
Введите выражение:
7+2-(2*(5^2))
Результат: -41.00
```

Приложение В.2 – Вычисление выражение

```
C:\Users\givap\source\repos\kursovoi\Debug\kursovoi.exe
2 - вычисления дополнительного кода;
0 - завершение программы.

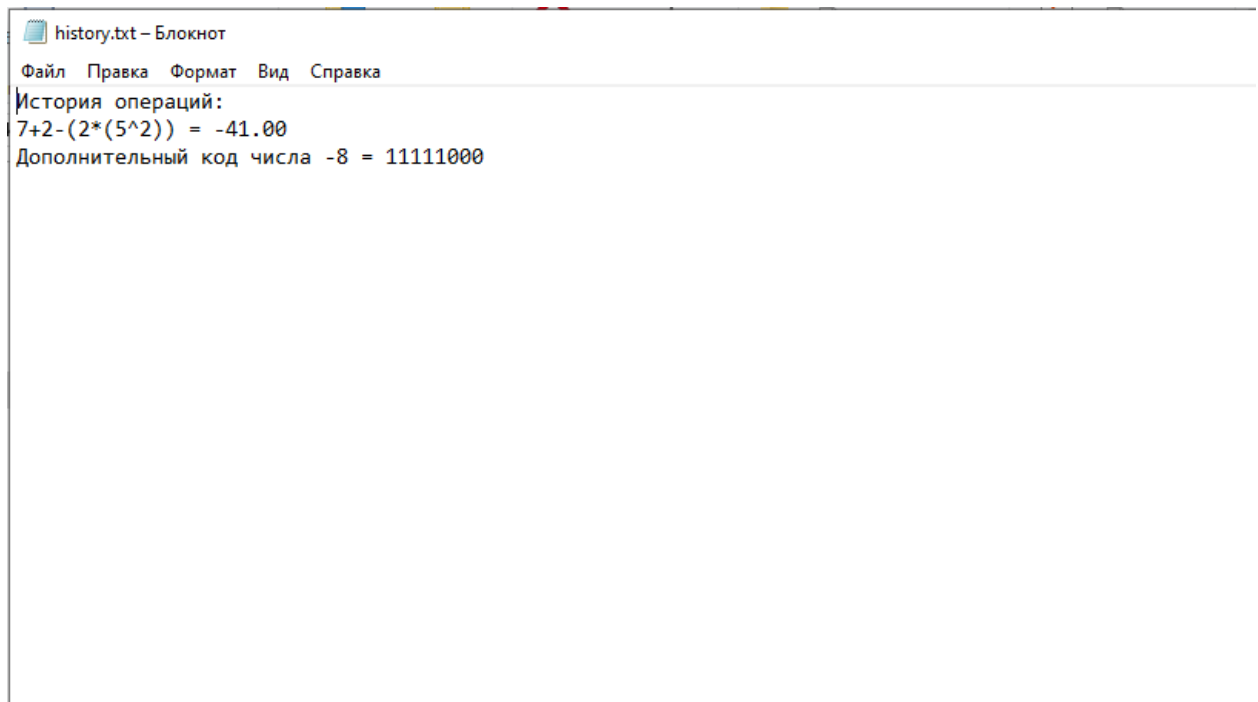
Список операций:
+ - операция сложения;
- - операция вычитания;
* - операция умножения;
/ - операция деления;
^ - операция возведения в степень;
$ - операция извлечения корня;
s - операция вычисления sin;
c - операция вычисления cos;
t - операция вычисления tg;
g - операция вычисления ctg.

Введите действие (1, 2, 0)
1
Введите выражение:
7+2-(2*(5^2))
Результат: -41.00
2
Введите десятичное число:
-8
Введите количество бит числа:
8
Десятичное число: -8
Дополнительный код: 11111000
■
```

Приложение В.3 – Вычисление дополнительного кода

Приложение С

Результат работы программы



```
history.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
История операций:
7+2-(2*(5^2)) = -41.00
Дополнительный код числа -8 = 11111000
```

Приложение С.3 – Файл history.txt