**ПРОЕКТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ**

1. **Разработка структуры приложения**

Для разработки приложения использовалась среда Microsoft Visual Studio 2019, включающая в себя широкий набор функциональных инструментов.

Для представления принципа работы программы ниже приведена ее

функциональная схема (рис. 1).

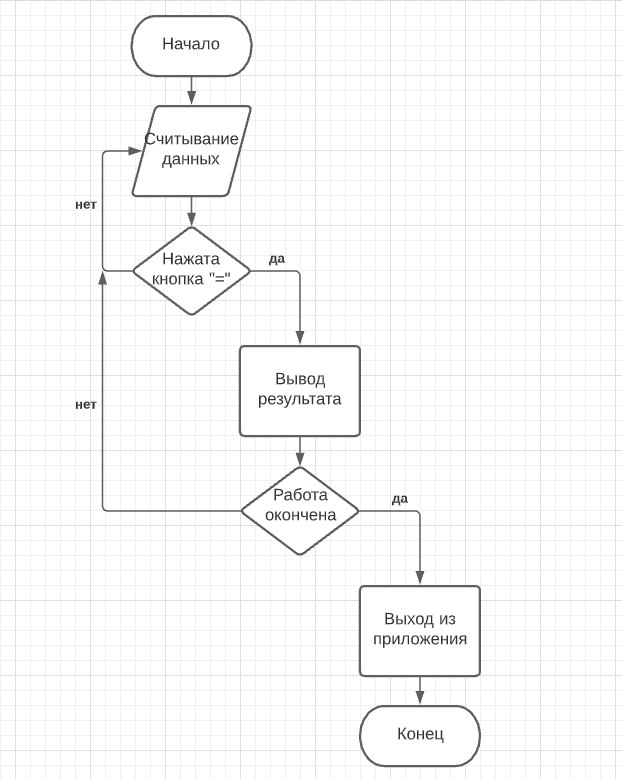


Рисунок 1. Функциональная схема программы «Калькулятор».

Пункт "Считывание данных" представляет собой часть программы, в которой пользователем вводятся числа и действия в ручную, и калькулятор их считывает.

В пункте "Нажата кнопка "=" ", при нажатии пользователем кнопки "=" калькулятор выведет результат, если же нажатия не произойдет, то калькулятор будет дальше считывать введенные данные.

После вывода результата пользователь может продолжить выполнять арифметические подсчеты, если это необходимо, или же завершить работу с приложением (пункт "Выход из приложения").

Общее назначение программного средства - выполнение арифметических и тригонометрических операций для использования в производственном, учебном процессах и повседневной жизни.

Реализуемая задача состоит в том, чтобы производилась корректная работа с тригонометрическими функциями, также чтобы при выборе действия выполнялась определенная операция, и имелась возможность сбросить полученный результат или же продолжить расчеты с данным результатом.

1. **Используемые функции Win32API**

Для создания программы "Калькулятор с расширенными возможностями", необходимо реализовать алгоритмы, позволяющие работать с арифметическими действиями, а также с тригонометрическими функциями. Необходимо организовать возможность сброса полученных результатов.

Блок-схемы алгоритмов представлены в Приложении 1.

Основные функции, реализуемые в программе из библиотеки User32.lib:

* LRESULT CALLBACK WndProc - производит обработку сообщений для окна (получает сообщения, которые Windows будет отсылать окну);
* GetModuleHandle - извлекает дескриптор указанного модуля;
* LoadCursor - загружает заданный ресурс курсора из исполняемого файла;
* LoadIcon - загружает указанный ресурс значка из исполняемого файла;
* RegisterClass - регистрирует класс окна для последующего использования при вызове функции CreateWindow;
* CreateWindow – создает окно;
* ShowWindow - устанавливает состояние показа определяемого окна;
* UpdateWindow - обновляет рабочую область заданного окна, отправляя сообщение WM\_PAINT окну, если регион обновления окна не пуст;
* GetMessage - извлекает сообщение из очереди сообщений вызывающего потока и помещает его в заданную структуру;
* TranslateMessage - переводит сообщения виртуальных клавиш в символьные сообщения;
* DispatchMessage - распределяет сообщение оконной процедуре;
* SendMessage - отправляет заданное сообщение окну;
* LOWORD - выделяет из 32-битового целочисленного значения младшее слово;
* HIWORD - выделяет из 32-битового целочисленного значения старшее слово;
* GetWindowText - копирует текст окна в буфер;
* SetWindowText - изменяет текст окна;
* PostQuitMessage - указывает системе, что поток сделал запрос на то, чтобы завершить свою работу (выйти). Это используется в ответ на сообщение WM\_DESTROY.
* DefWindowProc - вызывается оконной процедурой по умолчанию, чтобы обеспечить обработку по умолчанию любого сообщения окна, которые приложение не обрабатывает. Эта функция гарантирует то, что обрабатывается каждое сообщение

1. **Разработка архитектуры приложения**

Приложение разрабатывалась по архитектуре многоуровневого шаблона.

Данное приложение использует многоуровневый архитектурный шаблон. Данный шаблон используется для структурирования программ, которые можно разложить на группы неких подзадач, находящихся на определенных уровнях абстракции. Каждый слой представляет службы для следующего, более высокого слоя. На рисунке 2 представлена схема приложения.

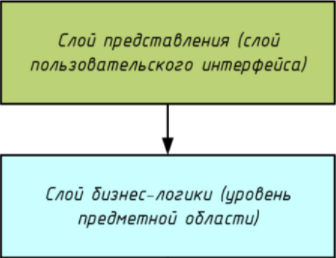


Рисунок. 2. Схема построения приложения по многоуровневому шаблону.

Слой представления в разрабатываемом приложении представлен в файле Calculator.cpp, в котором указаны следующие параметры:

* Главное окно. Расположение кнопок, поля ввода-вывода и параметрами главного окна;
* Иконка приложения при сборке проекта;

Слой бизнес-логики описан основным исполняемым файлом Calculator.cpp и файлом заголовков resource.h. Исполняемый файл Calculator.cpp состоит из функции обработки нажатия клавиш. Файл заголовков состоит из логики вычислений и вывода на экран.

1. **Разработка алгоритмов приложения**

Основной алгоритм приложения – непосредственное вычисление арифметических и тригонометрических выражений и последующий вывод на экран результата вычислений.

Для этого используется функция TCHAR\* Calculate(TCHAR\* buf) , где buf – это глобальный строчный массив символов, который был заполнен пользователем числами и знаками с помощью предыдущей функции.

Сперва, определяются приоритеты знаков, с помощью встроенной функции “map” из библиотеки “map”.

После определения приоритетов, необходимо перенести данные из символьного массива buf[256] в строку basic\_string, а после этого в строку потока basic\_stringstream.

Далее, используя basic\_stringstream, символы поочередно переводятся в числа типа double. После получения чисел, данная функция определяет какой знак действия ввел пользователь, и в зависимости от этого выполняются необходимые вычисления.

Последний шаг в функции – это вывод на экран вычисленного выражения.



Рисунок 3. Блок-схема функции Calculate.

1. **Разработка интерфейса взаимодействия пользователя с системой**

Основная и самая главная часть приложения – создание окна и дальнейшее взаимодействие с ним.

Для этого используется функция

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) , где

hWnd – дескриптор окна,

message – сообщение, посылаемое операционной системе,

wParam, lParam – параметры сообщений, для последующего обращения.

В данной функции взаимодействие с окном происходит с помощью множественного условного оператора Switch.

В оператор передается переменная message, в зависимости от значения которой будут выполняться определенные действия.

При message = WM\_CREATE, создается интерфейс взаимодействия пользователя с приложением (окно, кнопки) с помощью встроенной функции CreateWindow.

Message = WM\_COMMAND становится при нажатии пользователем на любую кнопку интерфейса, далее, в зависимости от того, какая именно кнопка была нажата, в программе выполняются определенные действия, например, ввод чисел, знаков, подсчитывание результата.

Если пользователь нажмет на крестик в правом верхнем углу программы, то переменной message присвоится значение WM\_DESTROY и произойдет удаление окна с экрана (программа закроется).

Message = default – это обработчик программы по умолчанию, для того чтобы обеспечить обработку любого сообщения окна, которое приложение не обрабатывает.

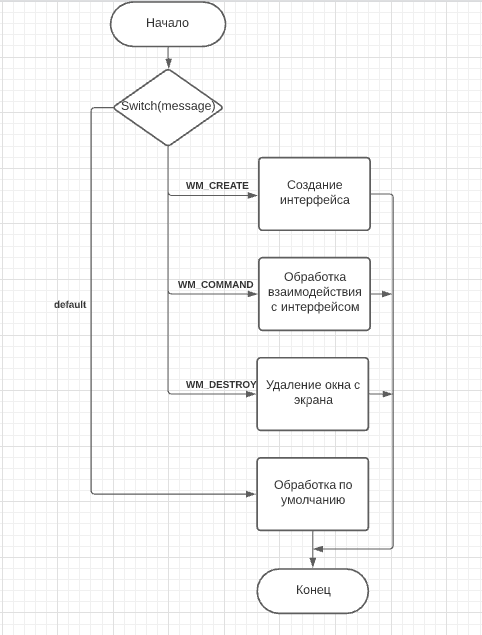


Рисунок 4. Блок-схема функции WndProc.