Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей Кафедра информатики

Дисциплина: Операционные среды и системное программирование

**ОТЧЕТ**

к лабораторной работе № 1

на тему «Основы программирования в Win 32 API. Оконное приложение Win 32 с минимальной достаточной функциональностью. Обработка основных оконных сообщений»

Выполнил:

студент гр. 153504

Хрищанович А.К.

Проверил:

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 19](#_Toc146622147)

[2 Краткие теоретические сведения 20](#_Toc146622148)

[3 Результаты выполнения лабораторной работы 22](#_Toc146622149)

[Выводы 24](#_Toc146622150)

[Список использованных источников 25](#_Toc146622151)

[Приложение А (обязательное) Листинг кода 26](#_Toc146622152)

## 1 Постановка задачи

Целью выполнения лабораторной работы является создание оконного приложения на Win32 API, обладающее минимальным функционалом, позволяющим отработать базовые навыки написания программы на Win32 API, использования виджетов и обработки оконных сообщений. Разработать калькулятор с графическим интерфейсом, способный выполнять математические операции, такие как вычисление синуса, косинуса, тангенса, котангенса, а также логарифма по основанию двойки.

## 2 Краткие теоретические сведения

Интерфейс прикладного программирования Win32 API представляет собой набор функций и подпрограмм, предоставляющих программный доступ к возможностям операционной системы Windows. Этот API предоставляет средства для управления и взаимодействия с операционной системой, а также для создания приложений, которые работают в среде Windows.

Win32 API включает в себя более 3000 функций, которые позволяют реализовать различные сервисы операционной системы. Эти функции могут быть вызваны из приложений для выполнения разнообразных задач, таких как создание процессов (например, CreateProcess), работа с файлами (например, CreateFile), и обработка оконных сообщений (например, GetMessage).

Для выполнения данной лабораторной работы, были использованы следующие теоретические сведения и концепции:

­– Win32 API (Application Programming Interface): Win32 API — это набор функций и подпрограмм, предоставляемых операционной системой Windows для взаимодействия с приложениями. Он предоставляет доступ к различным функциональным возможностям Windows, таким как создание окон, обработка сообщений, работа с файлами и др. В данной лабораторной работе Win32 API использован для создания графического интерфейса калькулятора и обработки событий.

– Обработка событий и сообщений: В Windows-приложениях взаимодействие с пользователем осуществляется через обработку событий и сообщений. В коде используются функции обработки сообщений, такие как SoftwareMainProcedure, которая отвечает за обработку событий, происходящих в приложении, таких как нажатия кнопок и меню.

– Создание пользовательского интерфейса: для создания графического интерфейса приложения были использованы стандартные элементы управления Windows, такие как окна, кнопки и текстовые поля. Эти элементы были созданы и настроены с использованием Win32 API функций.

– Математические операции: для выполнения математических операций, таких как сложение, вычитание, умножение и деление, а также расчет синуса, косинуса, тангенса и котангенса, использовались стандартные функции библиотеки cmath (например, sin, cos, tan) и операции с плавающей точкой для сохранения и обработки чисел.

– Работа с окнами и элементами управления: для создания оконного приложения была использована функция CreateWindow, которая создает окно с указанными характеристиками, такими как размер, положение и стиль.

Также использовались функции для работы с элементами управления, такие как кнопки и текстовые поля.

– Обработка меню: для создания меню и обработки команд меню использовалась функция CreateMenu и связанные с ней функции для добавления пунктов меню и обработки выбора команд.

– Обработка ошибок: для обработки ошибок, таких как деление на ноль или некорректный ввод, использовались функции вывода сообщений с помощью MessageBoxA.

– Интерфейс пользователя: Графический интерфейс приложения разработан с учетом стандартных принципов пользовательского интерфейса, таких как расположение элементов, цветовая схема и размещение текста.

## 3 Результаты выполнения лабораторной работы

В ходе выполнения лабораторной работы был разработан калькулятор. При запуске программы пользователь видит базовый интерфейс калькулятора, которым может воспользоваться. Так же абсолютно для каждого введенного числа пользователь может получить значения синуса, косинуса, тангенса, котангенса (значения во время подсчета переводятся в радианы) и логарифма по основанию двойки. Результат работы программы показан на рисунке 3.1.

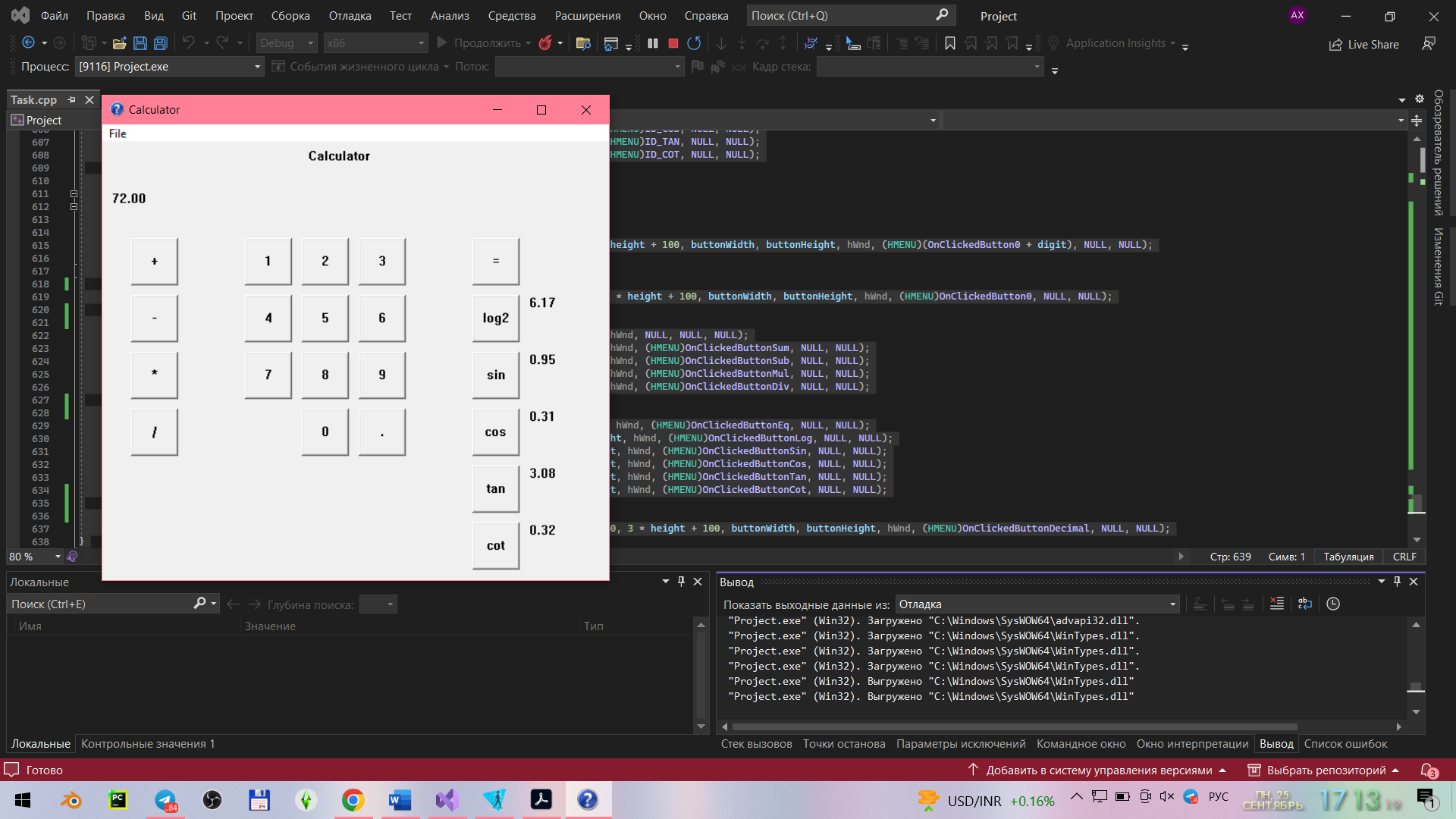


Рисунок 3.1 – Результат работы программы

Так же пользователь имеет доступ к меню, при помощи которого может очистить каждое поле графического интерфейса, а также выйти из программы. Интерфейс мен показан на рисунке 3.2.

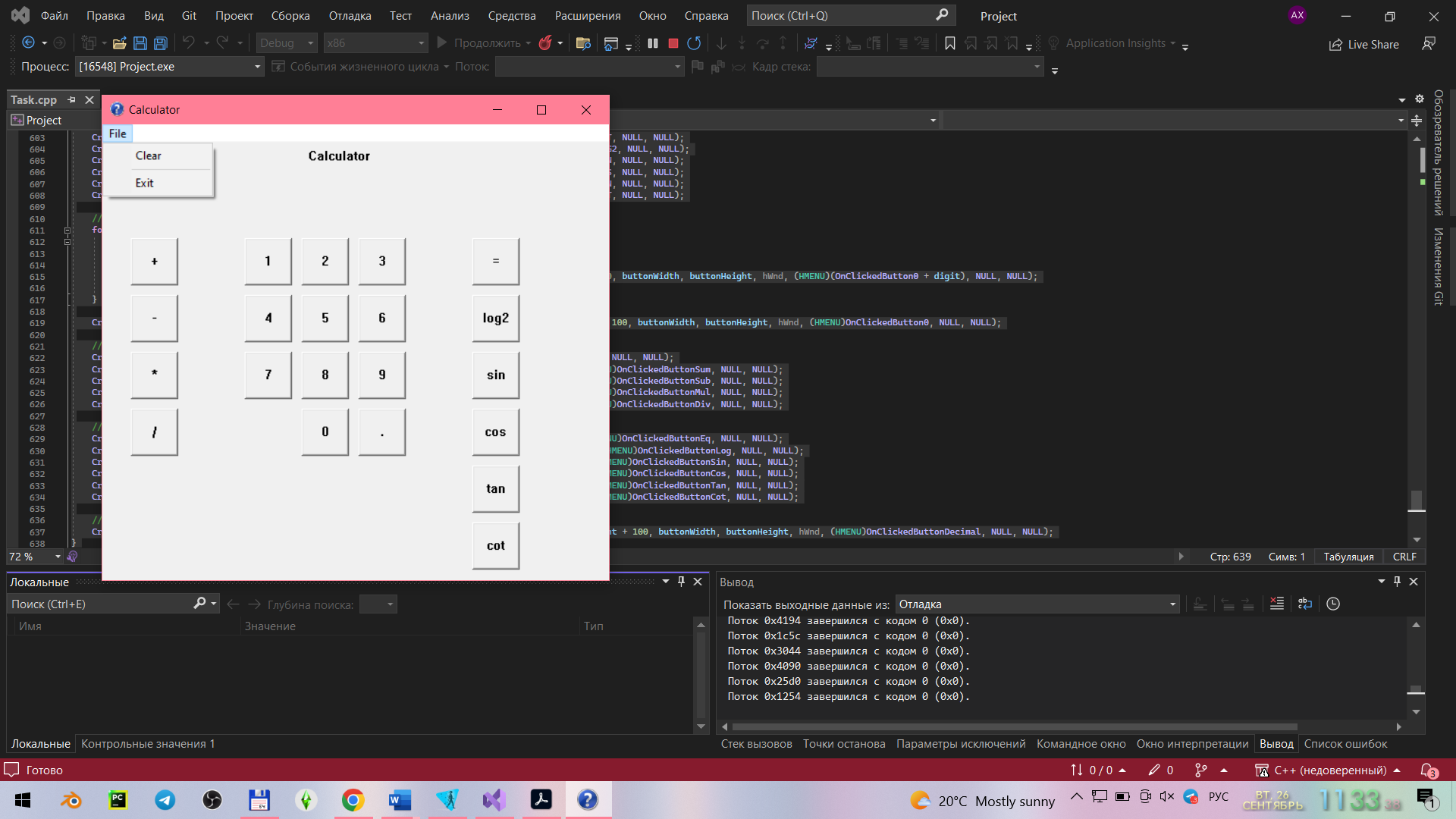


Рисунок 3.2 – Интерфейс меню

## ВЫВОДЫ

В ходе выполнения лабораторной работы были изучены основы программирования в Win 32 API. Как результат, создано оконное приложение с минимальной достаточной функциональностью и реализована обработка основных оконных сообщений.  Разработан калькулятор с графическим интерфейсом, способный выполнять математические операции, такие как вычисление синуса, косинуса, тангенса, котангенса, а также логарифма по основанию двойки.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Щупак Ю. Win32 API. Разработка приложений для Windows. – СПб: Питер, 2008. – 592 с.: ип.

[2] [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/api – Дата доступа 20.09.2023

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

**Task.cpp**

#include <Windows.h>

#include <iostream>

#include <cmath>

#define PI 3.14159265

#define OnMenuAction1 10

#define OnExitSoftware 11

#define ID\_EDIT 12

#define ID\_LOG2 19

#define ID\_SIN 20

#define ID\_COS 21

#define ID\_TAN 22

#define ID\_COT 23

#define OnClickedButtonLog 24

#define OnClickedButtonSin 25

#define OnClickedButtonCos 26

#define OnClickedButtonTan 27

#define OnClickedButtonCot 28

#define OnClickedButton1 1

#define OnClickedButton2 2

#define OnClickedButton3 3

#define OnClickedButton4 4

#define OnClickedButton5 5

#define OnClickedButton6 6

#define OnClickedButton7 7

#define OnClickedButton8 8

#define OnClickedButton9 9

#define OnClickedButton0 0

#define OnClickedButtonMul 13

#define OnClickedButtonDiv 14

#define OnClickedButtonSum 15

#define OnClickedButtonSub 16

#define OnClickedButtonEq 17

#define OnClickedButtonDecimal 18

LRESULT CALLBACK SoftwareMainProcedure(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp);

WNDCLASS NewWindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, HICON Icon, LPCWSTR Name, WNDPROC Procedure);

void MainWndAddMenu(HWND hWnd);

void MainWndAddWidget(HWND hWnd);

double tempValue = 0.0;

double currentValue = 0.0;

double storedValue = 0.0;

double decimalMultiplier = 1;

char currentOperator = '\0';

bool isNewInput = true;

bool decimalMode = false;

char buffer[50];

// Точка входа в приложение

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR args, int ncmdshow)

{

WNDCLASS SoftwareMainClass = NewWindowClass((HBRUSH)COLOR\_WINDOW, LoadCursor(NULL, IDC\_HAND), hInst, LoadIcon(NULL, IDI\_QUESTION),

L"MainWndClass", SoftwareMainProcedure);

if (!RegisterClassW(&SoftwareMainClass)) { return -1; }

MSG SoftwareMainMessage = { 0 };

CreateWindow(L"MainWndClass", L"Calculator", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 100, 100, 550, 520, NULL, NULL, NULL, NULL);

while (GetMessage(&SoftwareMainMessage, NULL, NULL, NULL))

{

TranslateMessage(&SoftwareMainMessage);

DispatchMessage(&SoftwareMainMessage);

}

return 0;

}

WNDCLASS NewWindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, HICON Icon, LPCWSTR Name, WNDPROC Procedure)

{

WNDCLASS NWC = { 0 };

NWC.hIcon = Icon;

NWC.hCursor = Cursor;

NWC.hInstance = hInst;

NWC.lpszClassName = Name;

NWC.hbrBackground = BGColor;

NWC.lpfnWndProc = Procedure;

return NWC;

}

LRESULT CALLBACK SoftwareMainProcedure(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp)

{

switch (msg)

{

case WM\_COMMAND:

switch (wp)

{

case OnMenuAction1:

// Сброс переменных до первоначального значения

currentValue = 0.0;

storedValue = 0.0;

decimalMultiplier = 1.0;

currentOperator = '\0';

isNewInput = true;

decimalMode = false;

// Сброс текста в текстовом поле (ID\_EDIT)

SetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), "");

SetWindowTextA(GetDlgItem(hWnd, ID\_LOG2), "");

SetWindowTextA(GetDlgItem(hWnd, ID\_SIN), "");

SetWindowTextA(GetDlgItem(hWnd, ID\_COS), "");

SetWindowTextA(GetDlgItem(hWnd, ID\_TAN), "");

SetWindowTextA(GetDlgItem(hWnd, ID\_COT), "");

break;

case OnExitSoftware:

PostQuitMessage(0);

break;

case OnClickedButton0:

{

int digit = 0;

if (isNewInput)

{

currentValue = digit;

isNewInput = false;

}

else

{

if (decimalMode)

{

// Если в режиме десятичных чисел, умножаем текущее значение на 10 и добавляем новую десятичную цифру.

currentValue = currentValue + digit \* decimalMultiplier;

decimalMultiplier \*= 0.1;

}

else

{

currentValue = currentValue \* 10 + digit;

}

}

// Обновляем текст в текстовом поле (ID\_EDIT) с текущим значением.

tempValue = currentValue;

sprintf\_s(buffer, sizeof(buffer), "%.2f", currentValue);

SetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), buffer);

}

break;

case OnClickedButton1:

{

int digit = 1;

if (isNewInput)

{

currentValue = digit;

isNewInput = false;

}

else

{

if (decimalMode)

{

currentValue = currentValue + digit \* decimalMultiplier;

decimalMultiplier \*= 0.1;

}

else

{

currentValue = currentValue \* 10 + digit;

}

}

tempValue = currentValue;

sprintf\_s(buffer, sizeof(buffer), "%.2f", currentValue);

SetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), buffer);

}

break;

case OnClickedButton2:

{

int digit = 2;

if (isNewInput)

{

currentValue = digit;

isNewInput = false;

}

else

{

if (decimalMode)

{

currentValue = currentValue + digit \* decimalMultiplier;

decimalMultiplier \*= 0.1;

}

else

{

currentValue = currentValue \* 10 + digit;

}

}

tempValue = currentValue;

sprintf\_s(buffer, sizeof(buffer), "%.2f", currentValue);

SetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), buffer);

}

break;

case OnClickedButton3:

{

int digit = 3;

if (isNewInput)

{

currentValue = digit;

isNewInput = false;

}

else

{

if (decimalMode)

{

currentValue = currentValue + digit \* decimalMultiplier;

decimalMultiplier \*= 0.1;

}

else

{

currentValue = currentValue \* 10 + digit;

}

}

tempValue = currentValue;

sprintf\_s(buffer, sizeof(buffer), "%.2f", currentValue);

SetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), buffer);

}

break;

case OnClickedButton4:

{

int digit = 4;

if (isNewInput)

{

currentValue = digit;

isNewInput = false;

}

else

{

if (decimalMode)

{

currentValue = currentValue + digit \* decimalMultiplier;

decimalMultiplier \*= 0.1;

}

else

{

currentValue = currentValue \* 10 + digit;

}

}

tempValue = currentValue;

sprintf\_s(buffer, sizeof(buffer), "%.2f", currentValue);

SetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), buffer);

}

break;

case OnClickedButton5:

{

int digit = 5;

if (isNewInput)

{

currentValue = digit;

isNewInput = false;

}

else

{

if (decimalMode)

{

currentValue = currentValue + digit \* decimalMultiplier;

decimalMultiplier \*= 0.1;

}

else

{

currentValue = currentValue \* 10 + digit;

}

}

tempValue = currentValue;

sprintf\_s(buffer, sizeof(buffer), "%.2f", currentValue);

SetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), buffer);

}

break;

case OnClickedButton6:

{

int digit = 6;

if (isNewInput)

{

currentValue = digit;

isNewInput = false;

}

else

{

if (decimalMode)

{

currentValue = currentValue + digit \* decimalMultiplier;

decimalMultiplier \*= 0.1;

}

else

{

currentValue = currentValue \* 10 + digit;

}

}

tempValue = currentValue;

sprintf\_s(buffer, sizeof(buffer), "%.2f", currentValue);

SetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), buffer);

}

break;

case OnClickedButton7:

{

int digit = 7;

if (isNewInput)

{

currentValue = digit;

isNewInput = false;

}

else

{

if (decimalMode)

{

currentValue = currentValue + digit \* decimalMultiplier;

decimalMultiplier \*= 0.1;

}

else

{

currentValue = currentValue \* 10 + digit;

}

}

tempValue = currentValue;

sprintf\_s(buffer, sizeof(buffer), "%.2f", currentValue);

SetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), buffer);

}

break;

case OnClickedButton8:

{

int digit = 8;

if (isNewInput)

{

currentValue = digit;

isNewInput = false;

}

else

{

if (decimalMode)

{

currentValue = currentValue + digit \* decimalMultiplier;

decimalMultiplier \*= 0.1;

}

else

{

currentValue = currentValue \* 10 + digit;

}

}

tempValue = currentValue;

sprintf\_s(buffer, sizeof(buffer), "%.2f", currentValue);

SetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), buffer);

}

break;

case OnClickedButton9:

{

int digit = 9;

if (isNewInput)

{

currentValue = digit;

isNewInput = false;

}

else

{

if (decimalMode)

{

currentValue = currentValue + digit \* decimalMultiplier;

decimalMultiplier \*= 0.1;

}

else

{

currentValue = currentValue \* 10 + digit;

}

}

tempValue = currentValue;

sprintf\_s(buffer, sizeof(buffer), "%.2f", currentValue);

SetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), buffer);

}

break;

case OnClickedButtonSum:

{

if (!isNewInput)

{

// Если уже есть введенное значение, сохраняем его и установливаем текущий оператор на сложение.

storedValue = currentValue;

currentOperator = '+';

isNewInput = true;

decimalMode = false;

decimalMultiplier = 1.0;

}

}

break;

case OnClickedButtonSub:

{

if (!isNewInput)

{

// -||- вычитание.

storedValue = currentValue;

currentOperator = '-';

isNewInput = true;

decimalMode = false;

decimalMultiplier = 1.0;

}

}

break;

case OnClickedButtonMul:

{

if (!isNewInput)

{

// -||- умножение

storedValue = currentValue;

currentOperator = '\*';

isNewInput = true;

decimalMode = false;

decimalMultiplier = 1.0;

}

}

break;

case OnClickedButtonDiv:

{

if (!isNewInput)

{

// -||- деление

storedValue = currentValue;

currentOperator = '/';

isNewInput = true;

decimalMode = false;

decimalMultiplier = 1.0;

}

}

break;

case OnClickedButtonEq:

{

// Обработка нажатия на кнопку "="

if (!isNewInput && currentOperator != '\0')

{

// Если есть введенное значение и текущий оператор, выполняем операцию.

switch (currentOperator)

{

case '+':

currentValue = storedValue + currentValue;

break;

case '-':

currentValue = storedValue - currentValue;

break;

case '\*':

currentValue = storedValue \* currentValue;

break;

case '/':

if (currentValue != 0)

{

currentValue = storedValue / currentValue;

}

else

{

// Обработка деления на ноль

MessageBoxA(hWnd, "Cannot divide by zero", "Error", MB\_OK | MB\_ICONERROR);

isNewInput = true;

break; // Выход из switch после обработки ошибки

}

break;

}

sprintf\_s(buffer, sizeof(buffer), "%.2f", currentValue);

SetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), buffer);

tempValue = currentValue;

currentValue = 0;

currentOperator = '\0'; // Сброс текущего оператора

decimalMode = false;

}

}

break;

case OnClickedButtonDecimal:

{

if (!decimalMode)

{

// Если еще не в режиме десятичных чисел, включаем его

decimalMode = true;

decimalMultiplier = 0.1;

}

// Добавляем десятичную запятую в текстовое поле

char currentText[50];

GetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), currentText, sizeof(currentText));

strcat\_s(currentText, sizeof(currentText), ".");

SetWindowTextA(GetDlgItem(hWnd, ID\_EDIT), currentText);

}

break;

case OnClickedButtonLog:

{

// log2

if (tempValue > 0)

{

double result = log2(tempValue);

sprintf\_s(buffer, sizeof(buffer), "%.2f", result);

SetWindowTextA(GetDlgItem(hWnd, ID\_LOG2), buffer);

}

else

{

// Обработка ошибки для отрицательных значений или нуля

MessageBoxA(hWnd, "Invalid input for log2", "Error", MB\_OK | MB\_ICONERROR);

}

}

break;

case OnClickedButtonSin:

{

// sin

double result = sin(tempValue \* PI / 180); // Преобразование угла в радианы

sprintf\_s(buffer, sizeof(buffer), "%.2f", result);

SetWindowTextA(GetDlgItem(hWnd, ID\_SIN), buffer);

}

break;

case OnClickedButtonCos:

{

// cos

double result = cos(tempValue \* PI / 180);

sprintf\_s(buffer, sizeof(buffer), "%.2f", result);

SetWindowTextA(GetDlgItem(hWnd, ID\_COS), buffer);

}

break;

case OnClickedButtonTan:

{

// tan

double result = tan(tempValue \* PI / 180);

sprintf\_s(buffer, sizeof(buffer), "%.2f", result);

SetWindowTextA(GetDlgItem(hWnd, ID\_TAN), buffer);

}

break;

case OnClickedButtonCot:

{

// cot

double result = 1.0 / tan(tempValue \* PI / 180);

sprintf\_s(buffer, sizeof(buffer), "%.2f", result);

SetWindowTextA(GetDlgItem(hWnd, ID\_COT), buffer);

}

break;

default: break;

}

break;

case WM\_CREATE:

MainWndAddMenu(hWnd);

MainWndAddWidget(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default: return DefWindowProc(hWnd, msg, wp, lp);

}

}

void MainWndAddMenu(HWND hWnd)

{

HMENU RootMenu = CreateMenu();

HMENU SubMenu = CreateMenu();

AppendMenu(SubMenu, MF\_STRING, OnMenuAction1, L"Clear");

AppendMenu(SubMenu, MF\_SEPARATOR, NULL, NULL);

AppendMenu(SubMenu, MF\_STRING, OnExitSoftware, L"Exit");

AppendMenu(RootMenu, MF\_POPUP, (UINT\_PTR)SubMenu, L"File");

SetMenu(hWnd, RootMenu);

}

void MainWndAddWidget(HWND hWnd)

{

// Размеры и позиции для кнопок

const int buttonWidth = 50;

const int buttonHeight = 50;

const int buttonPadding = 10; // Расстояние между кнопками

const int height = buttonHeight + buttonPadding;

const int width = buttonWidth + buttonPadding;

// Создание графического интерфейса

CreateWindowA("edit", "", WS\_VISIBLE | WS\_CHILD | ES\_READONLY, 10, 50, 460, 40, hWnd, (HMENU)ID\_EDIT, NULL, NULL);

CreateWindowA("edit", "", WS\_VISIBLE | WS\_CHILD | ES\_READONLY, 450, 160, 50, 50, hWnd, (HMENU)ID\_LOG2, NULL, NULL);

CreateWindowA("edit", "", WS\_VISIBLE | WS\_CHILD | ES\_READONLY, 450, 220, 50, 50, hWnd, (HMENU)ID\_SIN, NULL, NULL);

CreateWindowA("edit", "", WS\_VISIBLE | WS\_CHILD | ES\_READONLY, 450, 280, 50, 50, hWnd, (HMENU)ID\_COS, NULL, NULL);

CreateWindowA("edit", "", WS\_VISIBLE | WS\_CHILD | ES\_READONLY, 450, 340, 50, 50, hWnd, (HMENU)ID\_TAN, NULL, NULL);

CreateWindowA("edit", "", WS\_VISIBLE | WS\_CHILD | ES\_READONLY, 450, 400, 50, 50, hWnd, (HMENU)ID\_COT, NULL, NULL);

// Создание кнопок для цифр 1-9

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

int digit = i \* 3 + j + 1;

char buttonText[2] = { '0' + digit, '\0' };

CreateWindowA("button", buttonText, WS\_CHILD | WS\_VISIBLE, j \* width + 150, i \* height + 100, buttonWidth, buttonHeight, hWnd, (HMENU)(OnClickedButton0 + digit), NULL, NULL);

}

}

CreateWindowA("button", "0", WS\_CHILD | WS\_VISIBLE, buttonPadding + buttonWidth + 150, 3 \* height + 100, buttonWidth, buttonHeight, hWnd, (HMENU)OnClickedButton0, NULL, NULL);

// Создание знаков мат. операций

CreateWindowA("static", "Calculator", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 5, 5, 490, 20, hWnd, NULL, NULL, NULL);

CreateWindowA("button", "+", WS\_VISIBLE | WS\_CHILD, 30, 100, buttonWidth, buttonHeight, hWnd, (HMENU)OnClickedButtonSum, NULL, NULL);

CreateWindowA("button", "-", WS\_VISIBLE | WS\_CHILD, 30, 160, buttonWidth, buttonHeight, hWnd, (HMENU)OnClickedButtonSub, NULL, NULL);

CreateWindowA("button", "\*", WS\_VISIBLE | WS\_CHILD, 30, 220, buttonWidth, buttonHeight, hWnd, (HMENU)OnClickedButtonMul, NULL, NULL);

CreateWindowA("button", "/", WS\_VISIBLE | WS\_CHILD, 30, 280, buttonWidth, buttonHeight, hWnd, (HMENU)OnClickedButtonDiv, NULL, NULL);

// Создание кнопок для вычисления синуса, косинуса, тангенса, котангенса.

CreateWindowA("button", "=", WS\_VISIBLE | WS\_CHILD, 390, 100, buttonWidth, buttonHeight, hWnd, (HMENU)OnClickedButtonEq, NULL, NULL);

CreateWindowA("button", "log2", WS\_VISIBLE | WS\_CHILD, 390, 160, buttonWidth, buttonHeight, hWnd, (HMENU)OnClickedButtonLog, NULL, NULL);

CreateWindowA("button", "sin", WS\_VISIBLE | WS\_CHILD, 390, 220, buttonWidth, buttonHeight, hWnd, (HMENU)OnClickedButtonSin, NULL, NULL);

CreateWindowA("button", "cos", WS\_VISIBLE | WS\_CHILD, 390, 280, buttonWidth, buttonHeight, hWnd, (HMENU)OnClickedButtonCos, NULL, NULL);

CreateWindowA("button", "tan", WS\_VISIBLE | WS\_CHILD, 390, 340, buttonWidth, buttonHeight, hWnd, (HMENU)OnClickedButtonTan, NULL, NULL);

CreateWindowA("button", "cot", WS\_VISIBLE | WS\_CHILD, 390, 400, buttonWidth, buttonHeight, hWnd, (HMENU)OnClickedButtonCot, NULL, NULL);

// Для десятичных точек

CreateWindowA("button", ".", WS\_CHILD | WS\_VISIBLE, buttonPadding + buttonWidth \* 2 + 160, 3 \* height + 100, buttonWidth, buttonHeight, hWnd, (HMENU)OnClickedButtonDecimal, NULL, NULL);

**}­**