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 Цели работы 3](#_Toc146554844)

[2 Краткие теоретические сведения 4](#_Toc146554845)

[3 Полученные результаты 16](#_Toc146554846)

[Выводы 17](#_Toc146554847)

# 1 Цели работы

1. Изучить основы программирования в Win 32 API.
2. Создать оконное приложение Win 32 с минимальной достаточной функциональностью.
3. Реализовать обработку основных оконных сообщений.
4. 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 (Application Programming Interface Win32 – Win32 API) – это программный интерфейс, который используется для управления этими операционными системами Windows. Win32 API состоит из набора функций и подпрограмм, предоставляющих программный доступ к возможностям операционной системы, т.е. программные интерфейсы приложений представляют собой наборы функций (в этот обобщенный термин мы включаем и подпрограммы), которые обеспечивают сервисы данного приложения. Win32 API содержит более 3000 функции для реализации всех видов сервисов операционной системы.

Некоторые термины в пользовательской и программной документации Windows в разных контекстах имеют разные значения. Например, слово служба может относиться к вызываемой в операционной системе стандартной подпрограмме, драйверу устройства или к обслуживающему процессу. Что именно означают те или иные термины, показано в следующем списке:

Функции Windows API. Документированные, вызываемые подпрограммы в WindowsAPI. Например, CreateProcess, CreateFile и GetMessage.

Собственные системные службы (или системные вызовы). Недокументированные базовые службы в операционной системе, вызываемые при работе в пользовательском режиме. Например, NtCreateUserProcess является внутренней службой, которую функция Windows CreateProcess вызывает для создания нового процесса.

Функции поддержки ядра (или подпрограммы). Подпрограммы внутри операционной системы Windows, которые могут быть вызваны только из режима ядра. Например, ExAllocatePoolWithTag является подпрограммой, вызываемой драйверами устройств для выделения памяти из системных динамически распределяемых областей Windows (называемых пулами).

Службы Windows. Процессы, запускаемые Диспетчером управления службами (Windowsservicecontrolmanager). Например, служба Диспетчер задач запускается в виде процесса, работающего в пользовательском режиме, в котором поддерживается команда at (аналогичная UNIX-командам at или cron).

Библиотеки DLL (dynamic-link libraries — динамически подключаемые библиотеки). Набор вызываемых подпрограмм, связанных вместе в виде двоичного файла, который может быть загружен в динамическом режиме приложениями, которые используют эти подпрограммы.

Листинг 1 **–** Код исходной программы:

# #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);

# } 3 Полученные результаты

Результат работы программы показан на рисунке 3.1.

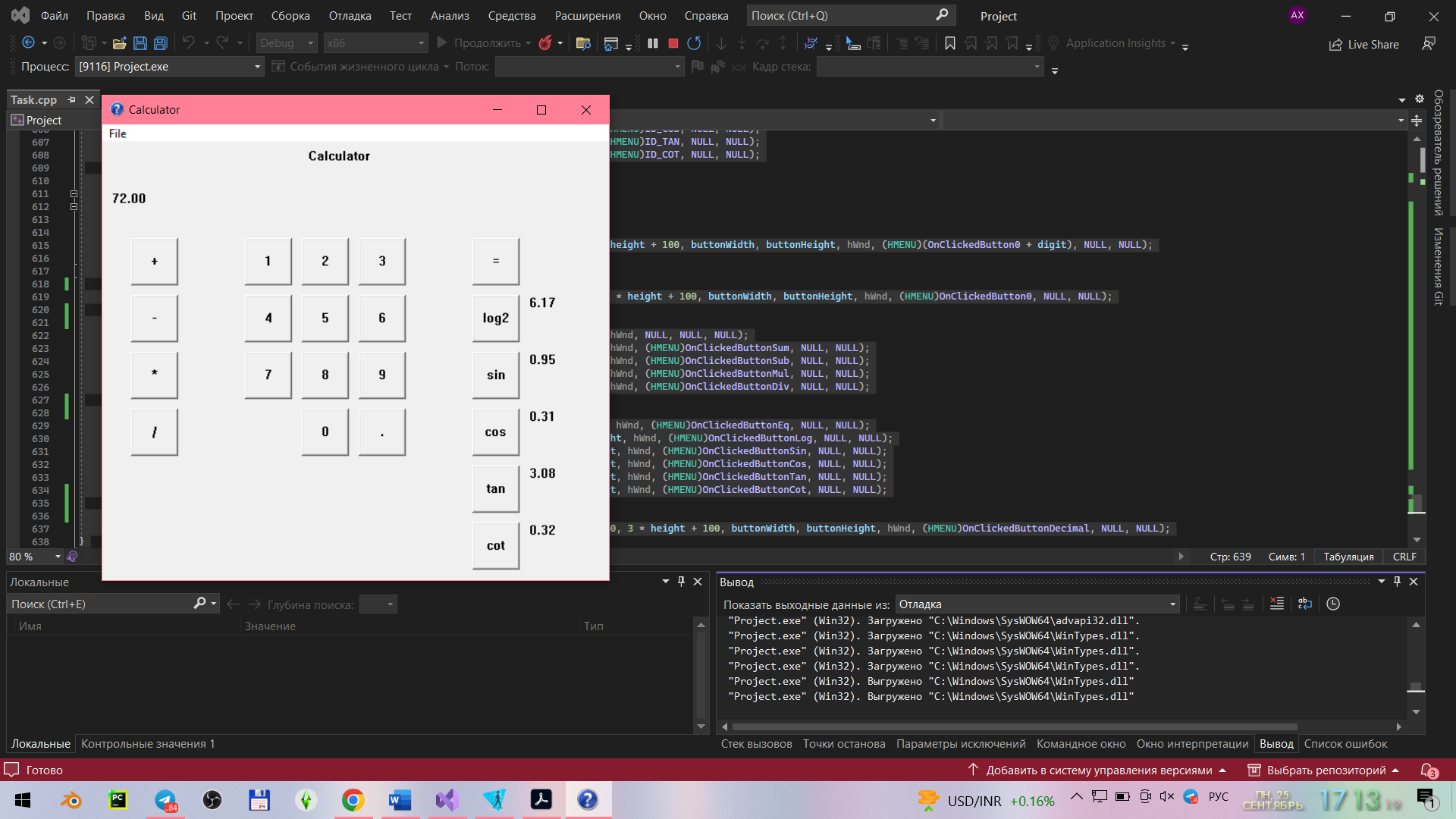


Рисунок 3.1 – Результат работы программы

# 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.