МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Вятский государственный университет»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

ЛАБОРАТОРНАЯ РАБОТА №3

«Разработка многооконного Windows-приложения с использованием API»

Отчёт по лабораторной работе дисциплины

«Технологии программирования»

Выполнил студент группы ИВТб-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Жеребцов К. А./

Проверил преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Долженкова М. Л./

Киров 2022

**Цель работы:** изучить базовые функции WinApi, а также сам принцип написания Windows-приложений.

**Задание:** с помощью функций WinAPI реализовать программу, в которой будет 2 окна: главное окно и 1 дочернее. На главном окне заполняется исходная матрица, скаляр, и здесь же выводится результирующая матрица. В дочернем окне пользователь выбирает умножение или деление матрицы на скаляр.

**Код программы:**

#include "Header.h"

static std::pair<bool, HWND> AddWindow(const std::wstring&& winClass, const std::wstring&& title,

HWND hParentWnd, const WNDPROC callback)

{

UnregisterClass(winClass.c\_str(), GetModuleHandle(NULL));

WNDCLASSEX wc{ sizeof(WNDCLASSEX) };

HWND hWindow{};

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hbrBackground = (HBRUSH)(COLOR\_WINDOW);

wc.hCursor = LoadCursor(NULL, IDC\_ARROW);

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wc.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

wc.lpfnWndProc = callback;

wc.lpszClassName = winClass.c\_str();

wc.style = CS\_VREDRAW | CS\_HREDRAW;

const auto create\_window = [&hWindow, &winClass, &title, &hParentWnd]()->std::pair<bool, HWND> {

if (hWindow = CreateWindow(winClass.c\_str(), title.c\_str(), WS\_OVERLAPPEDWINDOW,

250, 250, 250, 250, hParentWnd, NULL, NULL, NULL); !hWindow)

return { false, NULL };

ShowWindow(hWindow, SW\_SHOWDEFAULT);

UpdateWindow(hWindow);

return { true, hWindow };

};

if (!RegisterClassEx(&wc))

return create\_window();

return create\_window();

}

int CALLBACK wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInst, PWSTR szCmdLine, int nCmdShow)

{

static MSG msg{};

static HWND hwnd{}, hChildWnd{};;

WNDCLASSEX wc{ sizeof(WNDCLASSEX) };

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hbrBackground = (HBRUSH)(COLOR\_WINDOW);

wc.hCursor = LoadCursor(NULL, IDC\_ARROW);

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wc.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

wc.hInstance = hInstance;

wc.lpfnWndProc = [](HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)->LRESULT

{

switch (uMsg)

{

case WM\_CREATE:

{

MainWndAddWidjets(hWnd);

}

return 0;

case WM\_COMMAND:

{

switch (wParam)

{

case OnButtonChoooseClick:

{

if (ff == true)

{

f = 0;

int k = 0;

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 10; j++)

{

char buf[8];

GetWindowTextA(hEdit[k], buf, 8);

if (atoi(buf) == 0)

{

a[i][j] = 0;

}

else

{

a[i][j] = atoi(buf);

};

k++;

}

}

char buf[8];

GetWindowTextA(hEditSC, buf, 8);

scal = atoi(buf);

wchar\_t buf1[8];

int num = 0;

for (int i = 0; i < 10; i++)

{

{

for (int j = 0; j < 10; j++)

{

swprintf\_s(buf1, L"%d", a[i][j]);

SetWindowText(hEdit[num], (LPCWSTR)buf1);

num++;

}

}

}

swprintf\_s(buf1, L"%d", scal);

SetWindowText(hEditSC, (LPCWSTR)buf1);

if (hChildWnd)

DestroyWindow(hChildWnd);

//EnableWindow(hwnd, false);

const auto [flag, hChild] = AddWindow(L"ChildClass", L"Child", hWnd, [](HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)->LRESULT

{

switch (uMsg)

{

case WM\_CREATE:

{

ChildWndAddWidjets(hWnd);

SetWindowText(Static2, L"");

EnableWindow(hwnd, false);

}

return 0;

case WM\_COMMAND:

{

switch (wParam)

{

case OnButtonBackClick:

{

EnableWindow(hwnd, true);

DestroyWindow(hChildWnd);

if (!hChildWnd)

MessageBox(0, L"ошибка", L"df", MB\_OK);

}

break;

case OnButtonMultiplyClick:

{

f = 1;

function(a, scal, f);

wchar\_t buf2[16];

int num = 0;

for (int i = 0; i < 10; i++)

{

{

for (int j = 0; j < 10; j++)

{

swprintf\_s(buf2, L"%d", a[i][j]);

SetWindowText(EditRes[num], buf2);

num++;

}

}

}

EnableWindow(hwnd, true);

DestroyWindow(hChildWnd);

if (!hChildWnd)

MessageBox(hWnd, L"ошибка", L"df", MB\_OK);

//EnableWindow(hwnd, true);

}

break;

case OnButtonDivideClick:

{

f = 2;

function(a, scal, f);

if (f == 0)

{

SetWindowText(Static2, L"Divising by 0!");

//MessageBox(hWnd, L"Division by 0!", L"ERROR!", MB\_OK);

break;

};

wchar\_t buf2[10];

int k = 0;

for (int i = 0; i < 10; i++)

{

{

for (int j = 0; j < 10; j++)

{

swprintf\_s(buf2, L"%d", a[i][j]);

SetWindowText(EditRes[k], buf2);

k++;

}

}

}

EnableWindow(hwnd, true);

DestroyWindow(hChildWnd);

if (!hChildWnd)

MessageBox(hWnd, L"ошибка", L"df", MB\_OK);

//EnableWindow(hwnd, true);

}

break;

}

break;

}

return 0;

case WM\_CLOSE:

{

ShowWindow(hwnd, SW\_SHOWDEFAULT);

}

return 0;

}

return DefWindowProc(hWnd, uMsg, wParam, lParam);

});

hChildWnd = hChild;

}

}

break;

case OnButtonClearClick:

{

int k = 0;

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 10; j++)

{

Clear(hEdit[k]);

k++;

}

}

Clear(hEditSC);

}

break;

case OnButtonOKClick:

{

UpdateWindow(hwnd);

char buf[8];

GetWindowTextA(EditNum, buf, 8);

number = atoi(buf);

if ((number < 2) || (number > 10))

{

SetWindowText(StaticEr, L"ERROR!");

ff = false;

}

else

{

ff = true;

SetWindowText(StaticEr, NULL);

int k = 0;

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 10; j++)

{

if ((i >= number) || (j >= number))

{

ShowWindow(hEdit[k], 0);

ShowWindow(EditRes[k], 0);

}

else

{

ShowWindow(hEdit[k], 1);

ShowWindow(EditRes[k], 1);

}

//hEdit[k] = CreateWindow(L"edit", NULL, WS\_VISIBLE | WS\_CHILD | WS\_BORDER,

// i \* 80 + 40, j \* 30 + 50, 50, 20, hWnd, NULL, NULL, NULL);

k++;

}

}

/\*k = 0;

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 10; j++)

{

//EditRes[k] = CreateWindow(L"edit", NULL, WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_READONLY,

// i \* 100 + 900, j \* 30 + 50, 90, 20, hWnd, NULL, NULL, NULL);

k++;

}

}\*/

}

break;

}

return 0;

}

break;

}

return 0;

case WM\_DESTROY:

{

PostQuitMessage(EXIT\_SUCCESS);

}

return 0;

}

return DefWindowProc(hWnd, uMsg, wParam, lParam);

};

wc.lpszClassName = L"MainClass";

wc.lpszMenuName = NULL;

wc.style = CS\_HREDRAW | CS\_VREDRAW;

if (!RegisterClassEx(&wc))

return EXIT\_FAILURE;

if (hwnd = CreateWindow(wc.lpszClassName, L"Main", WS\_OVERLAPPEDWINDOW,

200, 200, 1500, 850, NULL, NULL, wc.hInstance, NULL); hwnd == INVALID\_HANDLE\_VALUE)

return EXIT\_FAILURE;

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

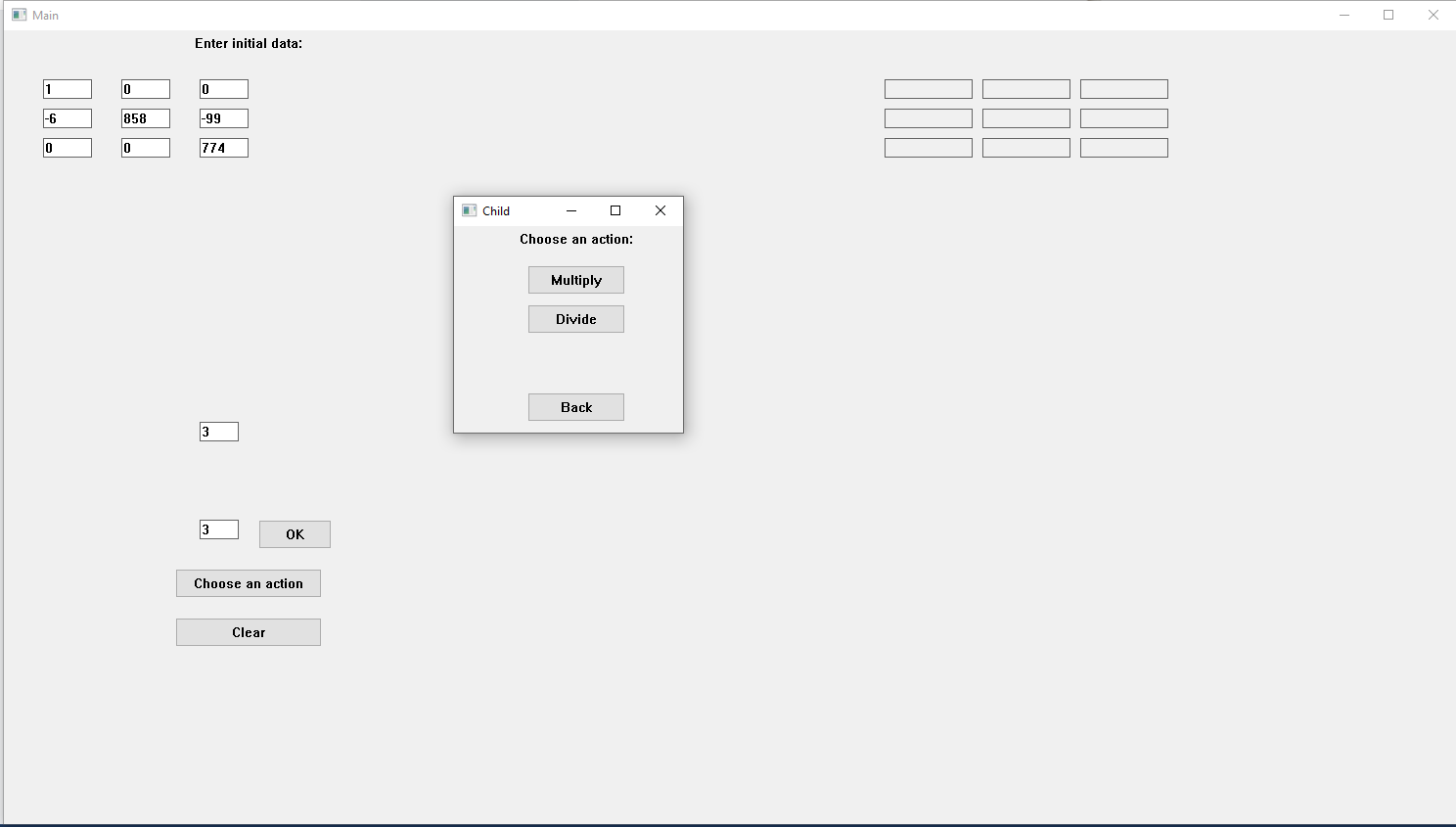
}

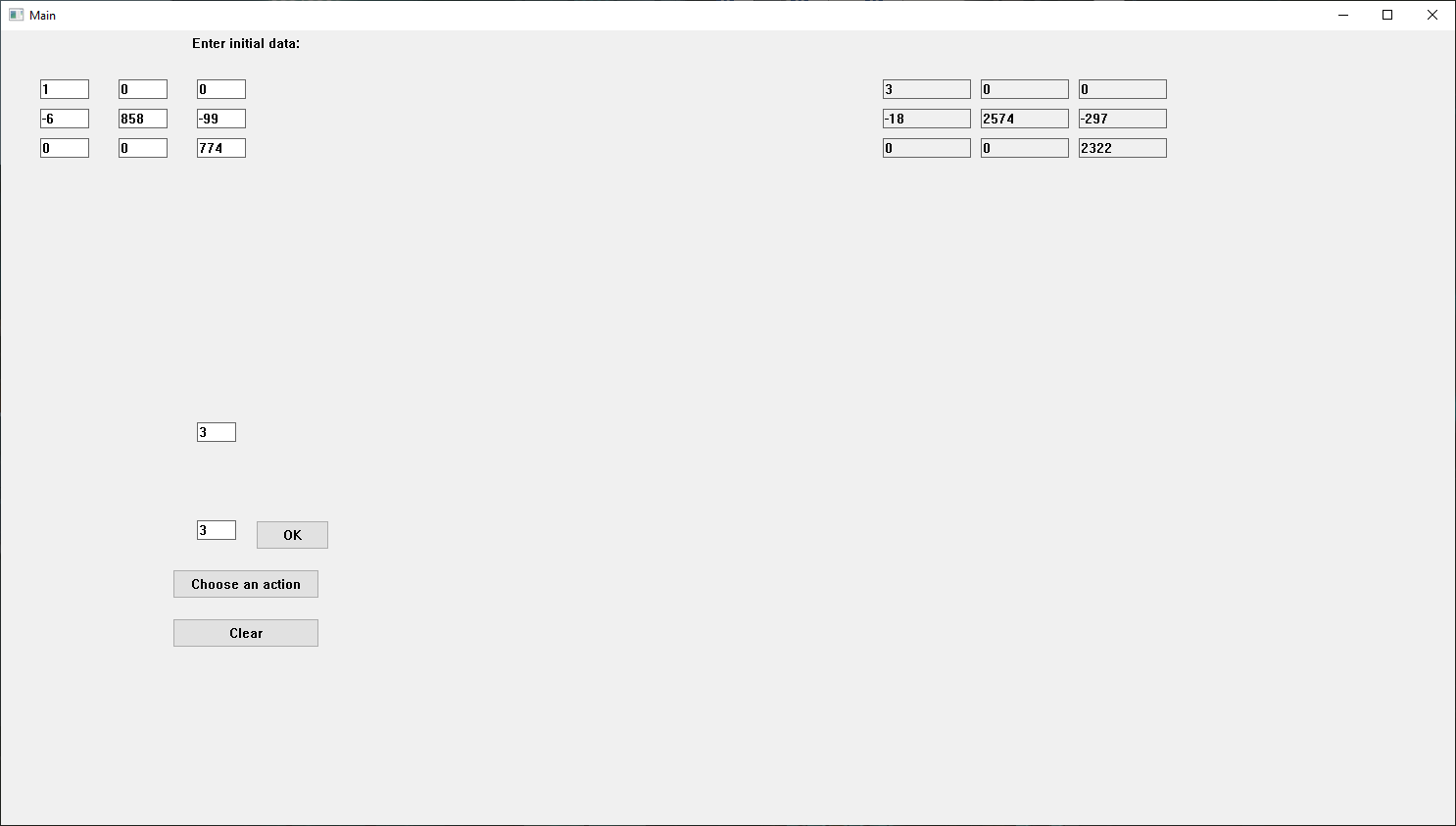
return static\_cast<int>(msg.wParam);

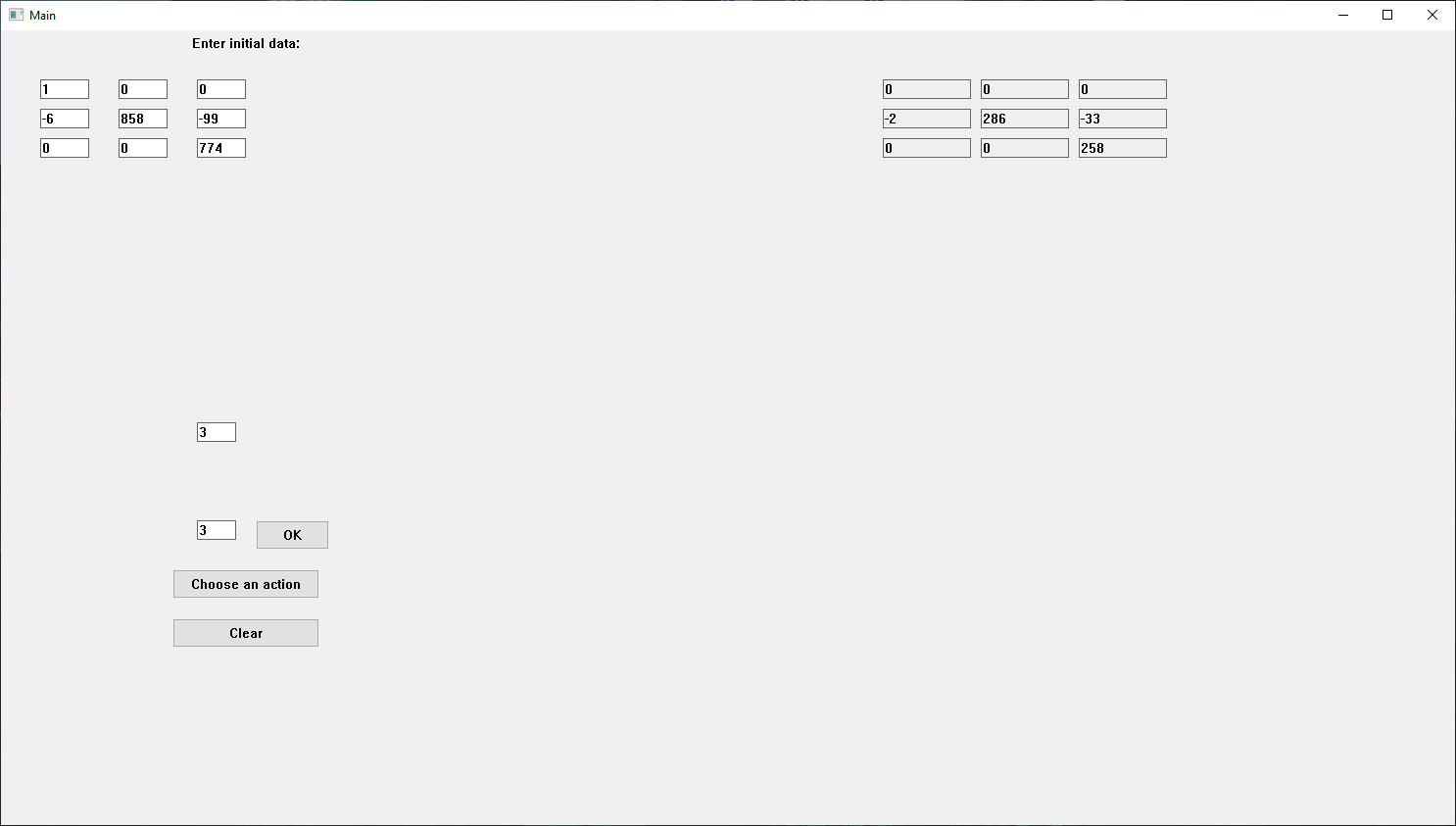
}

**Экранные формы программы:**









**Вывод:**

Таким образом, в результате выполнения данной лабораторной работы было написано многооконное Windows-приложение на языке С++ с помощью функций WinApi. Приложение состоит из двух окон главного и дочернего. На главном окне заполняется исходная матрица, скаляр, и здесь же выводится результирующая матрица. В дочернем окне пользователь выбирает умножение или деление матрицы на скаляр.

Для написания Windows-приложения вначале требуется подключить WinApi (#include <windows.h>).

В WinApi есть предопределенные классы окон, например EDIT(поле ввода), BUTTON(кнопка) или STATIC(Метка), для создания этих базовых элементов достаточно вызвать процедуру CreateWindow, только обязательно в стиле окна указать WS\_CHILD, а также передать дескриптор родительского окна. При необходимости можно указать дескриптор меню – это обычное число, идентификатор данного окна, это может понадобится при обработке сообщения WM\_COMMAND, чтобы через wparam узнать id окна, отправившего сообщение.

Для создания полноценного дочернего окна, достаточно просто создать и зарегистрировать еще один класс окна, и также указать, что оно дочернее при создании окна. Все элементы управления, вроде EDIT и BUTTON, являются дочерними. Создать также процедуру обработки сообщений дочернего окна, а дальше можно работать с ним как с полноценным окном.

Из всего вышеперечисленного можно сделать вывод, что были закреплены базовые знания при работе с WinApi, изучены базовые функции, а также сам принцип написания Windows-приложений,