Министерство образования и науки Российской Федерации Федеральное агентство по образованию Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №3

по курсу «Технология программирования»

# «Разработка многооконного приложения средствами WinAPI»

Выполнил студент группы ИВТ-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Птахова А.М/

Проверила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Долженкова М.Л/

Киров 2022

**1. Задание:** Разработать многооконное Windows приложение для работы с неориентированным графом с возможностью ввода количества вершин графа, указанием длины путей между вершинами и назначением конечной вершины – на основном окне программы, и с возможностью выбрать одну из двух операций (нахождение количества путей или кратчайшего пути) – на дочернем. Приложение должно быть реализовано на основе функций WinApi. В процессе разработки необходимо изучить особенность работы с событиями и сообщениями.

**2. Листинг кода**

#include <Windows.h>

#include <string>

#include <algorithm>

#include <iostream>

#include <wchar.h>

using namespace std;

#pragma comment(linker,"\"/manifestdependency:type='win32' \

name='Microsoft.Windows.Common-Controls' version='6.0.0.0' \

processorArchitecture='\*' publicKeyToken='6595b64144ccf1df' language='\*'\"")

#define OnButtonChoooseClick 1

#define OnButtonClearClick 2

#define OnButtonBackClick 3

#define OnButtonMultiplyClick 4

#define OnButtonDivideClick 5

#define OnButtonOKClick 6

#define OnButtonOK2Click 7

const int N = 5;

const CHAR\* ACCEPTED\_CHARS = ("0123456789+-");

//Main

HWND Static;

HWND ButtonChoose; //выбрать действие

HWND ButtonOk2;

HWND ButtonClear; //очитстить матрицу

HWND hEdit[10 \* 10]; //матрица

HWND EditRes1; //матрица результатат

HWND EditRes2;

HWND EditVer;

HWND EditNum; //размер матрицы

HWND ButtonOK; //применить

HWND StaticEr;

//Child

HWND Static1;

HWND Static2;

HWND ButtonBack;

HWND ButtonDivide;

HWND ButtonMultiply;

int a[6][6]; //матрица

int scal = 0; //число

int d[6]; // минимальное расстояние

int v[6]; // посещенные вершины

int co;

int temp, minindex ;

int begin\_index = 0;

int number = 6; //число

int past = 0;

int con;

int f = 0; //действие

bool ff = true;

void Clear(HWND edit)

{

SetWindowText(edit, L"");

}

void MainWndAddWidjets(HWND hWnd)

{

Static = CreateWindow(L"static", L"Enter top B:", WS\_VISIBLE | WS\_CHILD | ES\_CENTER,

-105, 300, 490, 20, hWnd, NULL, NULL, NULL);

EditVer = CreateWindow(L"edit", NULL, WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_NUMBER,

200, 300, 40, 20, hWnd, NULL, NULL, NULL);

//ButtonOk2 = CreateWindow(L"button", L"OK", WS\_CHILD | WS\_VISIBLE,

//250, 295, 75, 30, hWnd, (HMENU)OnButtonOK2Click, NULL, NULL);

Static = CreateWindow(L"static", L"Enter initial data:", WS\_VISIBLE | WS\_CHILD | ES\_CENTER,

5, 80, 490, 20, hWnd, NULL, NULL, NULL);

int k = 0;

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 6; j++)

{

if (i == j)

{

hEdit[k] = CreateWindow(L"edit", L"0", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_NUMBER | ES\_READONLY,

i \* 80 + 50, j \* 30 + 100, 50, 20, hWnd, NULL, NULL, NULL);

k++;

}

else if (i < j)

{

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

i \* 80 + 50, j \* 30 + 100, 50, 20, hWnd, NULL, NULL, NULL);

k++;

}

else

{

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

i \* 80 + 50, j \* 30 + 100, 50, 20, hWnd, NULL, NULL, NULL);

k++;

}

}

}

ButtonChoose = CreateWindow(L"button", L"Choose an action", WS\_CHILD | WS\_VISIBLE,

175, 550, 150, 30, hWnd, (HMENU)OnButtonChoooseClick, NULL, NULL);

ButtonClear = CreateWindow(L"button", L"Clear", WS\_CHILD | WS\_VISIBLE,

175, 600, 150, 30, hWnd, (HMENU)OnButtonClearClick, NULL, NULL);

Static = CreateWindow(L"static", L"Number of paths:", WS\_VISIBLE | WS\_CHILD | ES\_CENTER,

600, 5, 490, 20, hWnd, NULL, NULL, NULL);

EditRes1 = CreateWindow(L"edit", NULL, WS\_VISIBLE | WS\_CHILD | WS\_BORDER |ES\_READONLY,

900, 5, 40, 20, hWnd, NULL, NULL, NULL);

Static = CreateWindow(L"static", L"Shortest way:", WS\_VISIBLE | WS\_CHILD | ES\_CENTER,

600, 50, 490, 20, hWnd, NULL, NULL, NULL);

EditRes2 = CreateWindow(L"edit", NULL, WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_READONLY,

900, 50, 40, 20, hWnd, NULL, NULL, NULL);

Static = CreateWindow(L"static", L"Enter matrix size:", WS\_VISIBLE | WS\_CHILD | ES\_CENTER,

-105, 40, 490, 20, hWnd, NULL, NULL, NULL);

EditNum = CreateWindow(L"edit", NULL, WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_NUMBER,

200, 40, 40, 20, hWnd, NULL, NULL, NULL);

ButtonOK = CreateWindow(L"button", L"OK", WS\_CHILD | WS\_VISIBLE,

260, 35, 75, 30, hWnd, (HMENU)OnButtonOKClick, NULL, NULL);

StaticEr = CreateWindow(L"static", NULL, WS\_VISIBLE | WS\_CHILD | ES\_CENTER,

200, 450, 690, 50, hWnd, NULL, NULL, NULL);

}

void MainWnd(HWND hWnd)

{

}

void MinWnd2(HWND hWnd)

{

}

void ChildWndAddWidjets(HWND hWnd)

{

Static1 = CreateWindow(L"static", L"Choose an action:", WS\_VISIBLE | WS\_CHILD | ES\_CENTER,

5, 5, 240, 20, hWnd, NULL, NULL, NULL);

ButtonBack = CreateWindow(L"button", L"Back", WS\_CHILD | WS\_VISIBLE,

75, 170, 100, 30, hWnd, (HMENU)OnButtonBackClick, NULL, NULL);

ButtonDivide = CreateWindow(L"button", L"Number of paths from A to B", WS\_CHILD | WS\_VISIBLE,

15, 80, 200, 30, hWnd, (HMENU)OnButtonDivideClick, NULL, NULL);

ButtonMultiply = CreateWindow(L"button", L"Shortest way", WS\_CHILD | WS\_VISIBLE,

15, 40, 200, 30, hWnd, (HMENU)OnButtonMultiplyClick, NULL, NULL);

Static2 = CreateWindow(L"static", L"", WS\_VISIBLE | WS\_CHILD | ES\_CENTER,

5, 120, 240, 20, hWnd, NULL, NULL, NULL);

}

void function(int(&mas)[6][6], int (&d)[6], int(&flag))

{

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

{

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

{

if (flag == 2) //поиск минимального пути

{

int min;

for (int i = 0; i < 6; i++)

{

d[i] = 10000;

v[i] = 1;

}

d[begin\_index] = 0;

// Шаг алгоритма

do {

minindex = 10000;

min = 10000;

for (int i = 0; i < 6; i++)

{ // Если вершину ещё не обошли и вес меньше min

if ((v[i] == 1) && (d[i] < min))

{ // Переприсваиваем значения

min = d[i];

minindex = i;

}

}

// Добавляем найденный минимальный вес

// к текущему весу вершины

// и сравниваем с текущим минимальным весом вершины

if (minindex != 10000)

{

for (int i = 0; i < 6; i++)

{

if (mas[minindex][i] > 0)

{

temp = min + mas[minindex][i];

if (temp < d[i])

{

d[i] = temp;

}

}

}

v[minindex] = 0;

}

} while (minindex < 10000);

}

if (flag == 1)

{

if (mas[i][j] != 0 && i>1)

{

co = co + 1;

}

}

}

}

}

#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;

//break;

case WM\_COMMAND:

{

switch (wParam)

{

case OnButtonChoooseClick:

{

if (ff == true) // разрешние на откытие дочернего окна

{

f = 0;

int k = 0;

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 6; j++)

{

char buf[8];

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

if (i >= j)

{

if (atoi(buf) == 0)

{

a[i][j] = 0;

}

else

{

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

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

};

};

k++;

}

}

wchar\_t buf1[8];

int num = 0;

for (int i = 0; i < 6; i++)

{

{

for (int j = 0; j < 6; j++)

{

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

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

num++;

}

}

}

char buf[8];

GetWindowTextA(EditVer, buf, 8);

con = atoi(buf);

if (con > number)

{

MessageBox(0, L"Number greater than number of top!", L"df", MB\_OK);

break;

};

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"������", L"df", MB\_OK);

}

break;

case OnButtonMultiplyClick:

{

Clear(EditRes1);

Clear(EditRes2);

f = 2;

function(a, d, f);

wchar\_t buf2[16];

int num = 0;

swprintf\_s(buf2, L"%d", d[con]);

SetWindowText(EditRes2, buf2);

EnableWindow(hwnd, true);

DestroyWindow(hChildWnd);

if (!hChildWnd)

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

//EnableWindow(hwnd, true);

}

break;

case OnButtonDivideClick:

{

f = 1;

function(a, d, f);

wchar\_t buf2[10];

int k = 0;

swprintf\_s(buf2, L"%d", co+1);

SetWindowText(EditRes1, buf2);

EnableWindow(hwnd, true);

DestroyWindow(hChildWnd);

if (!hChildWnd)

MessageBox(hWnd, L"������", 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 < 6; i++)

{

for (int j = 0; j < 6; j++)

{

Clear(hEdit[k]);

k++;

}

}

}

break;

case OnButtonOKClick:

{

//UpdateWindow(hwnd);

number = 0;

char buf[8];

GetWindowTextA(EditNum, buf, 8);

number = atoi(buf);

if ((number < 2) || (number > 6))

{

SetWindowText(StaticEr, L"ERROR!");

ff = false;

}

else

{

ff = true;

UpdateWindow(hwnd);

MainWnd(hwnd);

SetWindowText(StaticEr, NULL);

int k = 0;

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 6; j++)

{

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

{

ShowWindow(hEdit[k], 0);

}

else

{

ShowWindow(hEdit[k], 1);

}

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, 1000, 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);

}

**3. Экранные формы**

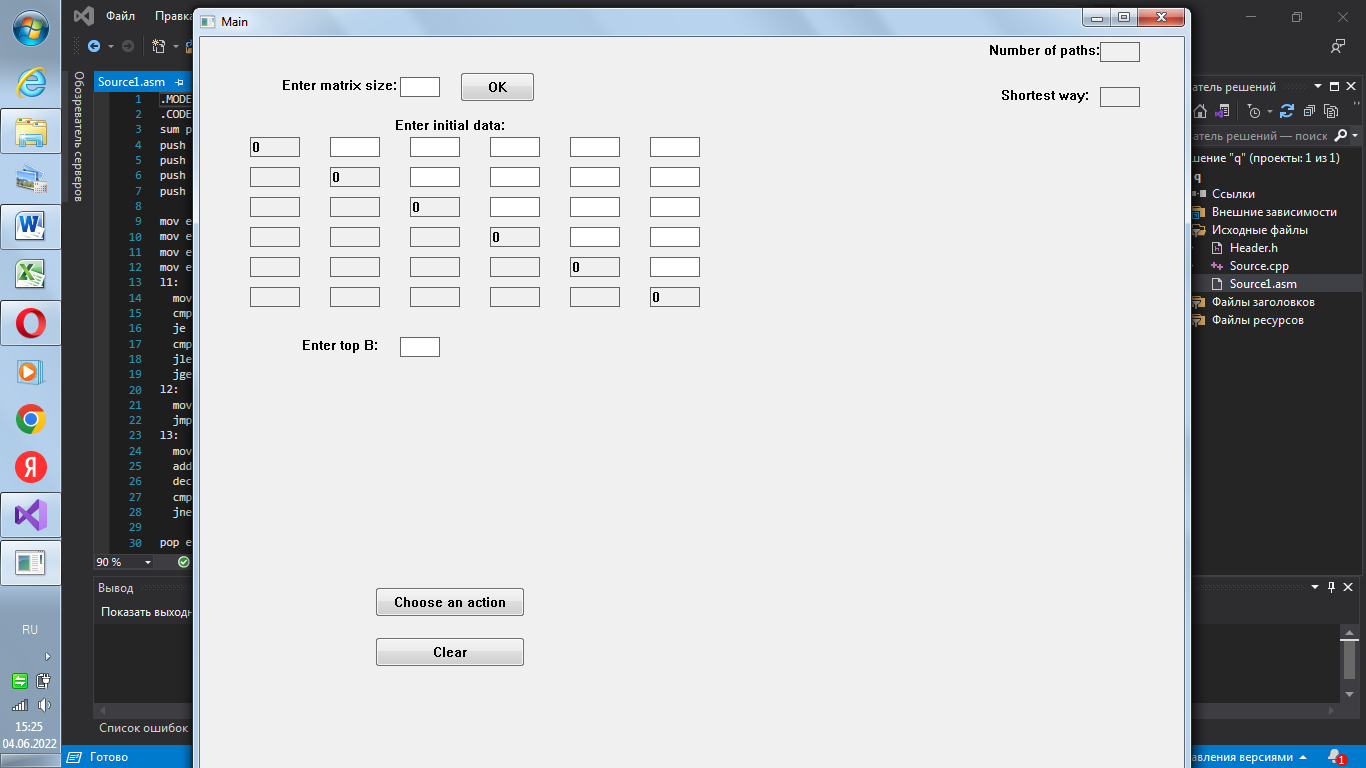


Рисунок 1 – главное окно в момент запуска программы

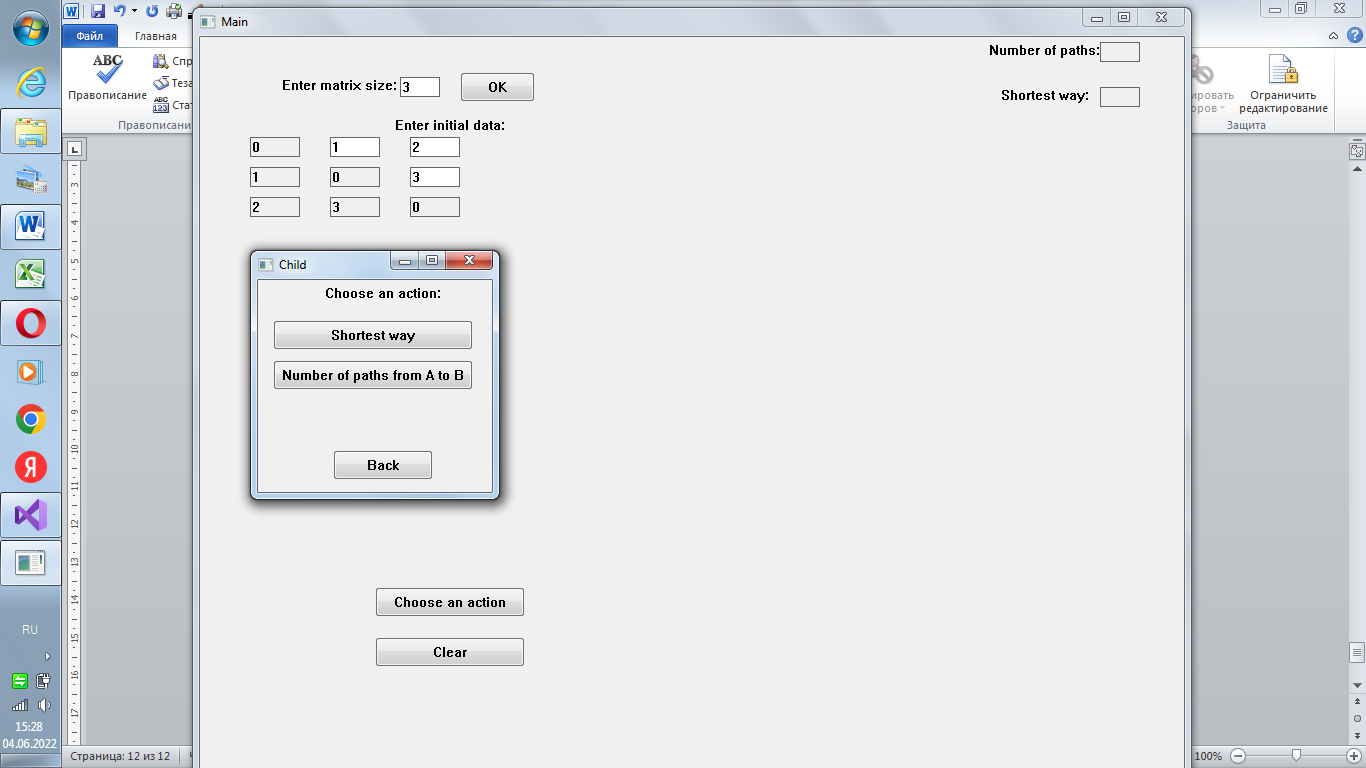


Рисунок 2 – дочернее окно

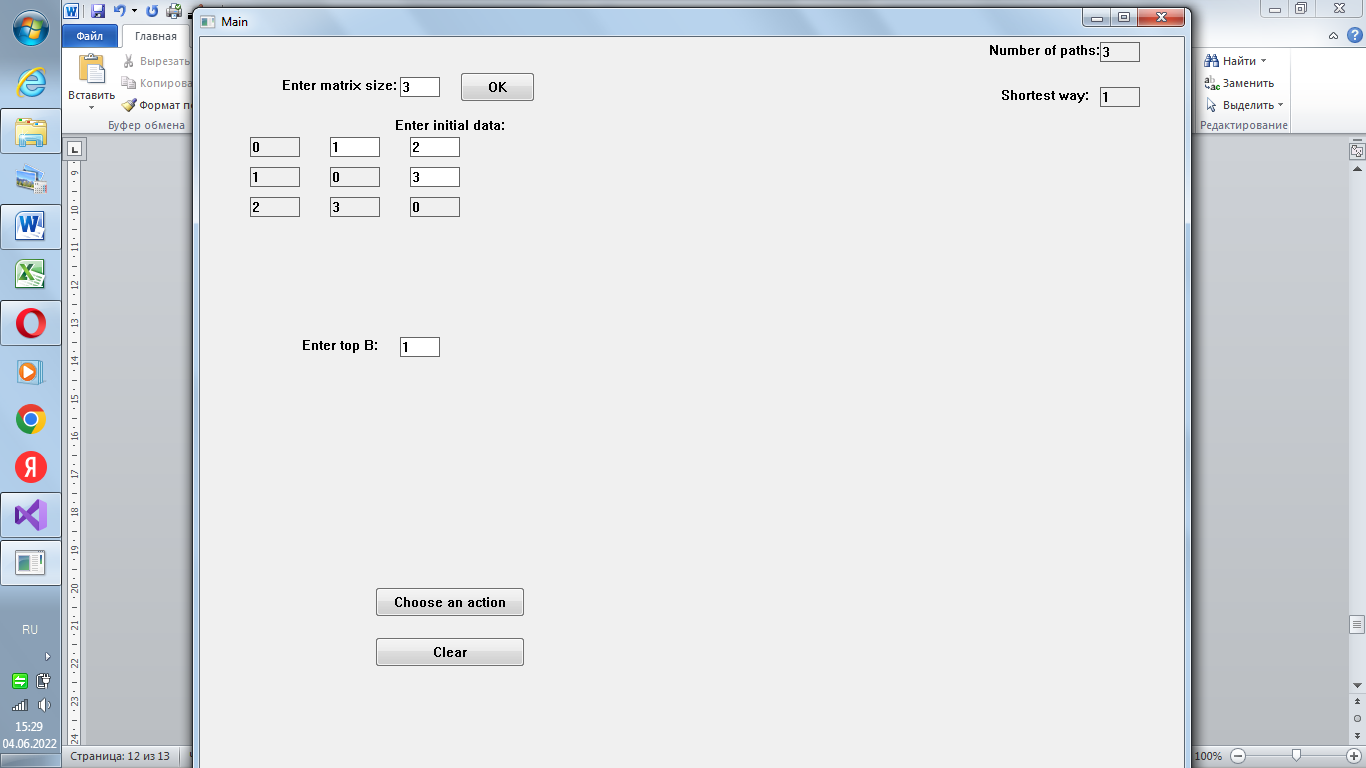


Рисунок 3 – главное окно после выполнения операций

**4. Вывод:**

В ходе выполнения лабораторной работы были изучены особенности работы с событиями и сообщениями при создании приложения на основе функций WinApi: любой компонент является окном, а любое действие над этим компонентом – отправкой сообщения.