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

Разработка многооконного приложение средствами WinAPI

Отчет по лабораторной работе №3 дисциплины

«Технологии программирования»

Выполнил студент группы ИВТ-22 /Крючков И. С/ Проверил /Долженкова М. Л./

Киров 2022

1. Задание

Написать программу с использованием WinApi для преобразования матрицы к треугольному виду и нахождения суммы элементов главной диагонали. На главном окне программы реализовать ввод матрицы, а также выбор необходимой операции – нахождение суммы элементов главной диагонали, либо преобразование матрицы к треугольному виду. В дочернем окне вывести результаты работы программы в соответствии с выбранной операцией.

Особенности

Windows-приложение состоит как минимум из двух основных процедур.

**WinMain()**, составляющей основу приложения. Данная функция является точкой входа в приложение. Выполняется при запуске программы, инициализирует все необходимые данные приложения, определяет его окна и запускает главное окно приложения. После создания главного окна запускается цикл обработки сообщений, извлекаемых из очереди. При получении сообщения вызывается процедура **wndProc()**. Представляет собой case структуру, позволяющую выбрать обработчик на основании id сообщения. Цикл обработки сообщения так же поддерживается на основе вызова winAPI.

1. Результаты работы программы

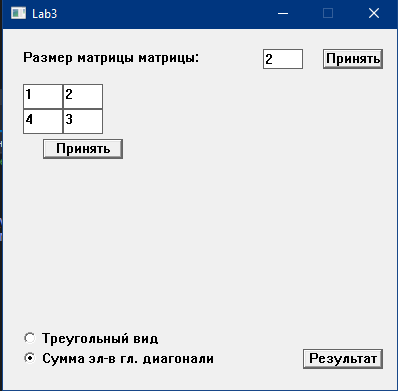


Рисунок 1 – Главное окно программы

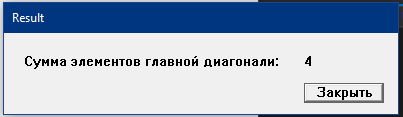


Рисунок 2 – Результат суммы элементов главной диагонали в дочернем окне

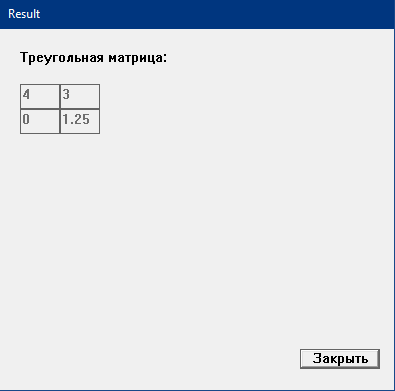


Рисунок 3 – Результат приведения матрицы к треугольному виду в дочернем окне

1. Листинг кода

#include <windows.h>

#include <windowsx.h>

#include <string>

#include <cmath>

#include <utility>

#include <sstream>

using namespace std;

HWND mainWindow, childWindow;

HWND sizeMatrixText, resultText, resultSumText;

HWND editSizeMatrix;

HWND submitButton1, submitButton2, radioButton1, radioButton2, resultButton, closeButton;

HWND matrixEdit[10][10], matrixEdit2[10][10];

LRESULT CALLBACK WndMainProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam);

LRESULT CALLBACK WndChildProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam);

// размеры главного окна

const int widthMainWnd = 410;

const int heightMainWnd = 400;

// размеры дочернего окна

const int widthChildWnd = 410;

const int heightChildWnd = 400;

const int heightChildWndSm = 120;

int m = 0;

int oldm;

double\*\* masMatrix;

char operationType = -1;

char matrState = 0;

void clearMatrix(double \*\*&matrx, int s) {

if (matrx != nullptr)

{

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

{

if (matrx[i] != nullptr)

{

delete[] matrx[i];

matrx[i] = nullptr;

}

}

delete[] matrx;

matrx = nullptr;

}

}

void copyMatrix(double\*\* a, double\*\* b, int n)

{

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

{

a[i] = new double[n];

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

{

a[i][j] = b[i][j];

}

}

}

char get\_trangle\_matrix(double\*\* matrx, double\*\*&a, int n) {

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

{

// поиск опорного элемента

int z = i;

char f = 0;

for (int h = z + 1; h < n; h++)

{

if (abs(a[z][i]) < abs(a[h][i]) || a[z][i] == 0)

{

if (a[z][i] == 0)

f = 1;

if (a[h][i] != 0)

swap(a[z], a[h]);

}

}

if (f == 1) {

return 1;

}

// прямой ход

for (int j = i + 1; j < n; j++)

{

double m = -a[j][i] / a[i][i];

for (int k = i; k < n; k++)

a[j][k] += a[i][k] \* m;

}

}

return 0;

}

double diagonal\_sum(double\*\* matrx, int n)

{

double sum = 0;

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

{

sum += matrx[i][i];

}

return sum;

}

// Точка входа

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdline, int nShowCmd)

{

MSG msg;

memset(&msg, 0, sizeof(msg));

// главное окно

WNDCLASSEX wndMain;

memset(&wndMain, 0, sizeof(wndMain));

wndMain.cbSize = sizeof(wndMain); // размер структуры

wndMain.style = CS\_VREDRAW | CS\_VREDRAW; // перерисовка при изменении размеров окна

wndMain.lpfnWndProc = (WNDPROC)WndMainProc; // обработчик окна

wndMain.hInstance = hInstance; // дескриптор экземпляра

wndMain.hbrBackground = (HBRUSH)COLOR\_WINDOW; // фон окна

wndMain.lpszClassName = L"MainWndClass";

if (!RegisterClassEx(&wndMain))

{

int nResult = GetLastError();

MessageBox(NULL, L"Класс главного окна не был создан!", L"Ошибка", MB\_ICONERROR);

}

mainWindow = CreateWindowEx(

NULL,

L"MainWndClass",

L"Lab3",

WS\_DLGFRAME | WS\_SYSMENU | WS\_MINIMIZEBOX,

CW\_USEDEFAULT, // x

0, // y

widthMainWnd, // width

heightMainWnd, // height

NULL, // id родительского окна

NULL,

hInstance,

NULL

);

if (!mainWindow)

{

MessageBox(NULL, L"Главное окно не было создано!", L"Ошибка", MB\_ICONERROR);

}

// дочернее окно

WNDCLASSEX wndChild;

memset(&wndChild, 0, sizeof(wndChild));

wndChild.cbSize = sizeof(wndChild); // размер структуры

wndChild.style = CS\_VREDRAW | CS\_VREDRAW; // перерисовка при изменении размеров окна

wndChild.lpfnWndProc = (WNDPROC)WndChildProc; // обработчик окна

wndChild.hInstance = hInstance; // дескриптор экземпляра

wndChild.hbrBackground = (HBRUSH)COLOR\_WINDOW; // фон окна

wndChild.lpszClassName = L"ChildWndClass";

if (!RegisterClassEx(&wndChild))

{

int nResult = GetLastError();

MessageBox(NULL, L"Класс дочернего окна не был создан!", L"Ошибка", MB\_ICONERROR);

}

childWindow = CreateWindowEx(

NULL,

L"ChildWndClass",

L"Result",

WS\_DLGFRAME,

CW\_USEDEFAULT, // x

0, // y

widthChildWnd, // width

heightChildWnd, // height

mainWindow, // id родительского окна

NULL,

hInstance,

NULL

);

if (!childWindow)

{

MessageBox(NULL, L"Дочернее окно не было создано!", L"Ошибка", MB\_ICONERROR);

}

ShowWindow(mainWindow, SW\_SHOW);

UpdateWindow(mainWindow);

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

LRESULT CALLBACK WndMainProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

{

unsigned int tLength;

char \*editValue = nullptr;

char \*err = NULL;

switch (message)

{

case WM\_CREATE:

sizeMatrixText = CreateWindow(L"STATIC", L"Размер матрицы матрицы:", WS\_VISIBLE | WS\_CHILD, 20, 20, 200, 20, hwnd, NULL, NULL, NULL);

editSizeMatrix = CreateWindow(L"EDIT", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_NUMBER, 260, 20, 40, 20, hwnd, NULL, NULL, NULL);

Edit\_LimitText(editSizeMatrix, 1); // ограничение кол-ва символов в поле ввода

submitButton1 = CreateWindow(L"BUTTON", L"Принять", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 320, 20, 60, 20, hwnd, (HMENU)1, NULL, NULL);

radioButton1 = CreateWindow(L"BUTTON", L"Треугольный вид", WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON,

20, 300, 150, 20, hwnd, (HMENU)3, NULL, NULL);

radioButton2 = CreateWindow(L"BUTTON", L"Сумма эл-в гл. диагонали", WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON,

20, 320, 200, 20, hwnd, (HMENU)4, NULL, NULL);

resultButton = CreateWindow(L"BUTTON", L"Результат", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 300, 320, 80, 20, hwnd, (HMENU)5, NULL, NULL);

break;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case 1: // принять размеры матрицы

oldm = m;

tLength = GetWindowTextLength(editSizeMatrix) + 1;

editValue = new char[tLength];

GetWindowTextA(editSizeMatrix, editValue, tLength + 1);

m = atoi(editValue);

delete[] editValue;

if (m < 1)

{

MessageBox(hwnd, L"Допустимое значение: от 1 до 9", L"Ошибка", MB\_OK);

return 0;

}

clearMatrix(masMatrix, oldm);

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

{

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

{

DestroyWindow(matrixEdit[i][j]);

}

}

DestroyWindow(submitButton2);

//Создание таблиц

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

{

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

{

matrixEdit[i][j] = CreateWindow(L"EDIT", L"0", WS\_VISIBLE | WS\_CHILD | WS\_BORDER,

j \* 40 + 20, i \* 25 + 55, 40, 25, hwnd, NULL, NULL, NULL);

Edit\_LimitText(matrixEdit[i][j], 3); // ограничение кол-ва символов в поле ввода

}

}

submitButton2 = CreateWindow(L"BUTTON", L"Принять", WS\_VISIBLE | WS\_CHILD | WS\_BORDER,

m \* 20, m \* 25 + 60, 80, 20, hwnd, (HMENU)2, NULL, NULL);

matrState = 0;

break;

case 2: // принять матрицу

if (matrState == 0)

{

try

{

masMatrix = new double\* [m];

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

{

masMatrix[i] = new double[m];

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

{

tLength = GetWindowTextLength(matrixEdit[i][j]) + 1;

editValue = new char[tLength];

GetWindowTextA(matrixEdit[i][j], editValue, tLength + 1);

masMatrix[i][j] = strtod(editValue, &err);

Edit\_Enable(matrixEdit[i][j], false);

if (\*err)

{

throw 100;

}

delete[] editValue;

}

}

SetWindowText(submitButton2, L"Изменить");

matrState = 1;

}

catch (...)

{

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

{

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

{

Edit\_Enable(matrixEdit[i][j], true);

}

}

MessageBox(hwnd, L"Вводите числа!", L"Ошибка", MB\_OK);

return -1;

}

}

else

{

clearMatrix(masMatrix, m);

matrState = 0;

SetWindowText(submitButton2, L"Принять");

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

{

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

{

Edit\_Enable(matrixEdit[i][j], true);

}

}

}

break;

case 3: // radio 1;

switch (Button\_GetCheck(radioButton1))

{

case BST\_CHECKED:

operationType = 0;

break;

}

break;

case 4: // radio 2;

switch (Button\_GetCheck(radioButton2))

{

case BST\_CHECKED:

operationType = 1;

break;

}

break;

case 5: // результаты

if (operationType == -1){

MessageBox(hwnd, L"Необходимо выбрать операцию", L"Ошибка", MB\_OK);

}

else if (matrState == 0)

{

MessageBox(hwnd, L"Необходимо принять матрицу", L"Ошибка", MB\_OK);

}

else

{

EnableWindow(mainWindow, false);

ShowWindow(childWindow, SW\_SHOW);

SetFocus(childWindow);

}

break;

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hwnd, message, wParam, lParam);

}

}

LRESULT CALLBACK WndChildProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_SHOWWINDOW:

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

{

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

{

DestroyWindow(matrixEdit2[i][j]);

}

}

DestroyWindow(closeButton);

if (operationType == 0){

double\*\* resMatrix = new double\* [m];

copyMatrix(resMatrix, masMatrix, m);

char trer = get\_trangle\_matrix(masMatrix, resMatrix, m);

if (trer == 0)

{

SetWindowPos(hwnd, NULL, 0, 0, widthChildWnd, heightChildWnd, SWP\_NOMOVE);

resultText = CreateWindow(L"STATIC", L"Треугольная матрица:", WS\_VISIBLE | WS\_CHILD, 20, 20, 300, 20, hwnd, NULL, NULL, NULL);

closeButton = CreateWindow(L"BUTTON", L"Закрыть", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 300, 320, 80, 20, hwnd, (HMENU)1, NULL, NULL);

//Создание таблицы

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

{

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

{

wstringstream wss;

wss << floor(resMatrix[i][j] \* 1000.0 + 0.5) / 1000.0;

matrixEdit2[i][j] = CreateWindow(

L"EDIT", wss.str().c\_str(), WS\_VISIBLE | WS\_CHILD | WS\_BORDER,

j \* 40 + 20, i \* 25 + 55, 40, 25, hwnd, NULL, NULL, NULL);

Edit\_Enable(matrixEdit2[i][j], false);

}

}

}

else

{

SetWindowPos(hwnd, NULL, 0, 0, widthChildWnd, heightChildWndSm, SWP\_NOMOVE);

resultText = CreateWindow(L"STATIC", L"Невозможно получить треугольную матрицу", WS\_VISIBLE | WS\_CHILD, 20, 20, 300, 20, hwnd, NULL, NULL, NULL);

closeButton = CreateWindow(L"BUTTON", L"Закрыть", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 300, 50, 80, 20, hwnd, (HMENU)1, NULL, NULL);

}

clearMatrix(resMatrix, m);

}

else

{

SetWindowPos(hwnd, NULL, 0, 0, widthChildWnd, heightChildWndSm, SWP\_NOMOVE);

resultText = CreateWindow(L"STATIC", L"Сумма элементов главной диагонали:", WS\_VISIBLE | WS\_CHILD, 20, 20, 300, 20, hwnd, NULL, NULL, NULL);

closeButton = CreateWindow(L"BUTTON", L"Закрыть", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, 300, 50, 80, 20, hwnd, (HMENU)1, NULL, NULL);

double resSum = diagonal\_sum(masMatrix, m);

wstringstream wss;

wss << resSum;

resultSumText = CreateWindow(L"STATIC", wss.str().c\_str(), WS\_VISIBLE | WS\_CHILD, 300, 20, 50, 20, hwnd, NULL, NULL, NULL);

}

break;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case 1: // закрыть

DestroyWindow(resultText);

ShowWindow(childWindow, SW\_HIDE);

EnableWindow(mainWindow, true);

SetFocus(mainWindow);

break;

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

break;

default:

return DefWindowProc(hwnd, message, wParam, lParam);

}

}

1. Вывод

В ходе выполнения лабораторной работы была разработана многооконная программа с использованием функций WinApi, реализующая ввод матрицы, преобразование матрицы к треугольному виду и нахождение суммы элементов главной диагонали.