МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

**«Изучение особенностей использования динамических библиотек»**

Отчет по лабораторной работе № 4 по дисциплине

«Технологии программирования»

Выполнил студент группы: ПИб-2301\_\_\_\_\_\_\_\_\_\_\_\_\_ /Морозов Д.А.

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

**Киров 2023**

**Цели работы:**

На основе задания лабораторной работы 3 модифицировать приложения.   
Создать 2 библиотечных модуля: Модуль пользовательского интерфейса и модуль вычислительных операций.  
Интерфейсный модуль подключить к основному приложению используя неявное связывание, а вычислительный - явное. Сделать выводы.

**Задание:**

* Добавить алгоритм вычисления минимального количества цветов для раскраски графа в DLL и подключить ее явно;
* Добавить процедуру создания интерфейса главного окна в DLL и подключить ее неявно.

**Листинг кода:**

WNDmain.h

#pragma once

#pragma comment(lib, "D:\\Documents\\GitHub\\prog-tech\\Lab4\\Debug\\Interface.lib")

#include<Windows.h>

#include<string>

#include "D:\Documents\GitHub\prog-tech\Lab4\Interface\interface.h";

extern "C" int graphColoring(int matrix[6][6], int numVertices, int\* numColors);

//макросы для облегчения работы

#define OnButtonClicked 1

#define TextBufferSize 50

#define DlgIndexNumberA 200

#define DlgIndexNumberB 210

char Buffer[TextBufferSize];

int readChar;

//беззнаковое целое число

unsigned numA;

unsigned numB;

//инициализация оконных переменных для передачи их значений между функциями

HWND childText;

HWND hEditControl;

HWND hStaticControl;

HWND hStaticControlNums;

HWND hWnd;

//инициализация необходимых глобальных переменных

int matrix[6][6];

std::string StrMat;

std::string path;

//конструктор класса главного окна

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

WNDCLASS NWC = { 0 };

NWC.hCursor = Cursor;

NWC.hIcon = Icon;

NWC.hInstance = hInst;

NWC.lpszClassName = Name;

NWC.hbrBackground = BGColor;

NWC.lpfnWndProc = Procedure;

return NWC;

}

//конструктор класса дочернего окна

WNDCLASS NewWindowChildClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, HICON Icon, LPCWSTR Name, WNDPROC ProcedureChild) {

WNDCLASS NWC = { 0 };

NWC.hCursor = Cursor;

NWC.hIcon = Icon;

NWC.hInstance = hInst;

NWC.lpszClassName = Name;

NWC.hbrBackground = BGColor;

NWC.lpfnWndProc = ProcedureChild;

return NWC;

}

//процедура преобразования текстовой матрицы в матрицу типа int

void ParseMatrix()

{

int k = 0;

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

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

if (StrMat[k] == '0') { matrix[i][j] = 0; };

if (StrMat[k] == '1') { matrix[i][j] = 1; };

k++;

}

}

}

//проверка корректности введенных значений

bool ValidateMatrix() {

if (StrMat.size() != 36) {

return false;

}

for (int i = 0; i < 36; i++) {

//если символ не 1 и не 0, то проверка не пройдена

if (StrMat[i] != 48 && StrMat[i] != 49) { return false; };

}

return true;

}

//процедура обработки сообщений главного окна

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

//typedef int(\*func)(int(\*)[6], int, int\*);

//HINSTANCE hinstDLL = LoadLibrary(TEXT("graphColoring.dll"));

//func graphColoring = (func)GetProcAddress(hinstDLL, "graphColoring");

switch (msg) {

case WM\_COMMAND:

switch (wp)

{

//при нажатии кнопки

case OnButtonClicked:

//получение значений из полей выбора вершин

readChar = GetWindowTextA(hEditControl, Buffer, TextBufferSize);

//замена переносов строк в текстовой матрице

StrMat = Buffer;

StrMat.erase(std::remove(StrMat.begin(), StrMat.end(), '\r'), StrMat.end());

StrMat.erase(std::remove(StrMat.begin(), StrMat.end(), '\n'), StrMat.end());

//запуск проверок корректности ввода

if (ValidateMatrix()) {

SetWindowTextA(hStaticControl, "Матрица задана корректно");

ParseMatrix();

}

else {

SetWindowTextA(hStaticControl, "Матрица задана некорректно!");

};

if (ValidateMatrix()) {

//выключаем главное окно

//это необходимо, так как дочернее окно должно быть модальным

EnableWindow(hWnd, FALSE);

int numColors = 0;

//if (hinstDLL == NULL) {

// MessageBox(hWnd, L"Библиотека graphColoring.dll не найдена!", L"Ошибка!", NULL);

// EnableWindow(hWnd, TRUE);

//}

//else {

int numVertices = 6;

int result = graphColoring(matrix, numVertices, &numColors);

path = "\nМинимальное кол-во цветов для раскраски графа - " + std::to\_string(numColors) + "\n";

//создается дочернее окно

HWND childWindow = CreateWindowEx(

0, L"ChildWndClass", L"Дочернее окно", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE,

300, 300, 300, 150, hWnd, NULL, NULL, NULL

);

//проверка на случай если есть проблемы с созданием дочернего окна

if (childWindow == NULL) {

MessageBox(childWindow, L"Не удалось создать дочернее окно", L"Ошибка", MB\_OK | MB\_ICONERROR);

return -1;

//}

};

}

break;

default: break;

}

break;

//при создании

case WM\_CREATE:

//процедура с созданием необходимых компонентов главного окна

hStaticControl = WndStaticControlChild(hWnd);

MainWndAddWidgetsChild(hWnd);

hEditControl = WndEditFromChild(hWnd);

break;

//при уничтожении

case WM\_DESTROY:

//FreeLibrary(hinstDLL);

PostQuitMessage(0);

break;

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

}

}

//процедура обработки сообщений дочернего окна

LRESULT CALLBACK SoftwareChildProcedure(HWND childWindow, UINT msg, WPARAM wp, LPARAM lp) {

switch (msg) {

case WM\_CREATE: {

//создание статического текста, куда будет записан разультат Дийкстры

if (!IsWindow(childText)) {

childText = CreateWindowEx(0, L"STATIC", L"", WS\_CHILD | WS\_VISIBLE,

0, 0, 300, 300, childWindow, NULL, NULL, NULL);

}

//вывод на дочернее окно ответа

SetWindowTextA(childText, path.c\_str());

break; }

//при закрытии

case WM\_CLOSE:

//включаем главное окно и удаляем дочернее

EnableWindow(hWnd, TRUE);

DestroyWindow(childWindow);

break;

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

}

}

graphColoring.cpp

#include "pch.h"

#define DLL\_EXPORT \_\_declspec(dllexport)

extern "C" {

DLL\_EXPORT int graphColoring(int matrix[6][6], int numVertices, int\* numColors)

{

// Инициализируем все вершины без цвета

int colors[6];

memset(colors, -1, sizeof(colors));

// Окрасим первую вершину первым цветом

colors[0] = 0;

\*numColors = 1;

// Окрашиваем оставшиеся вершины, используя жадный алгоритм

for (int v = 1; v < numVertices; v++) {

// Найдем доступные цвета для этой вершины

bool available[6];

memset(available, true, sizeof(available));

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

if (matrix[v][i] && colors[i] != -1) {

available[colors[i]] = false;

}

}

// Выбираем первый доступный цвет

int chosenColor;

for (chosenColor = 0; chosenColor < numVertices; chosenColor++) {

if (available[chosenColor]) {

break;

}

}

// Если доступного цвета не найдено, вернем -1 (неверный ввод)

if (chosenColor == numVertices) {

return -1;

}

// Окрашиваем вершину выбранным цветом

colors[v] = chosenColor;

// Обновляем минимальное количество необходимых цветов

if (chosenColor + 1 > \*numColors) {

\*numColors = chosenColor + 1;

}

}

// Возвращаем окрашенный граф

int result = 0;

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

result |= (colors[i] << (i \* 3));

}

return result;

}

}

graphColoring.h

#pragma once

extern "C" {

\_\_declspec(dllimport) int graphColoring(int matrix[6][6], int numVertices, int\* numColors);

}

interface.cpp

// MyDLL.cpp

#include "pch.h"

#define DLL\_EXPORT1 \_\_declspec(dllexport)

#define DLL\_EXPORT2 \_\_declspec(dllexport)

#define DLL\_EXPORT3 \_\_declspec(dllexport)

extern "C++" {

DLL\_EXPORT1 HWND WndEditFromChild(HWND hWnd) {

HWND hEditControl = CreateWindowA("edit", "000000\r\n000000\r\n000000\r\n000000\r\n000000\r\n000000", WS\_VISIBLE | WS\_CHILD | ES\_MULTILINE | ES\_NUMBER, 0, 30, 55, 100, hWnd, NULL, NULL, NULL, NULL);

return hEditControl;

}

DLL\_EXPORT2 HWND WndStaticControlChild(HWND hWnd) {

HWND hStaticControl = CreateWindowA("static", "Введите матрицу размером 6x6", WS\_VISIBLE | WS\_CHILD, 0, 0, 490, 20, hWnd, NULL, NULL, NULL, NULL);

return hStaticControl;

}

DLL\_EXPORT3 void MainWndAddWidgetsChild(HWND hWnd) {

CreateWindowA("button", "Посчитать", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 200, 140, 100, 30, hWnd, (HMENU)1, NULL, NULL, NULL);

}

}

interface.h

#pragma once

#ifdef \_\_cplusplus

extern "C++" {

#endif

#include <Windows.h>

#ifdef \_\_cplusplus

}

#endif

\_\_declspec(dllimport) HWND WndEditFromChild(HWND hWnd);

\_\_declspec(dllimport) HWND WndStaticControlChild(HWND hWnd);

\_\_declspec(dllimport) void MainWndAddWidgetsChild(HWND hWnd);

}

**Экранные формы:**

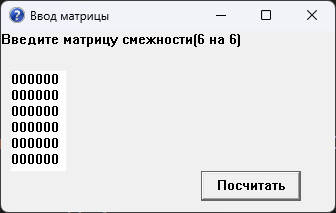


Рис.1 – главное окно при запуске

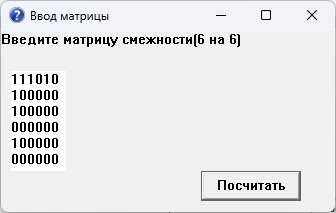


Рис. 2 – главное окно с заданными значениями

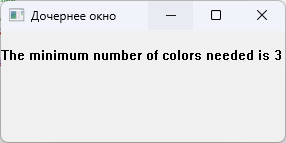


Рис.3 – дочернее окно с ответом

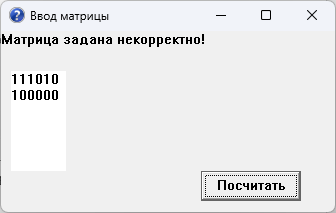


Рис. 4 – обработка некорректно ввода матрицы

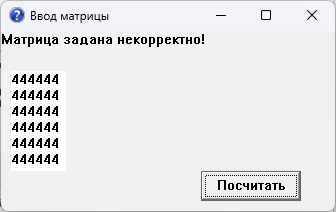


Рис. 5 – обработка некорректно ввода матрицы

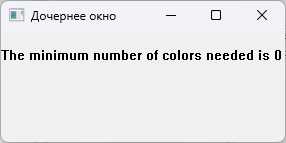


Рис. 6 – ответ в случае, когда матрица заполнена нулями (вершин нет)

**Вывод:**

В рамках данной лабораторной работы были созданы две динамические библиотеки, которые вносят изменения в оконное приложение, разработанное в процессе выполнения третьей лабораторной работы.

Одна из библиотек содержит функцию, реализующую жадный алгоритм для определения минимального количества цветов, необходимых для раскраски графа. Эта библиотека подключается явным образом во время выполнения программы.

Вторая библиотека содержит процедуры, создающие компоненты WinAPI, которые подключаются неявным образом при запуске программы.

При сравнении методов подключения динамических библиотек стоит отметить, что каждый из них имеет свои преимущества и может быть полезен в различных ситуациях. Например, явное подключение целесообразно, если библиотека содержит большое количество функций, из которых требуется использовать лишь некоторые. Это позволяет сэкономить ресурсы, подключая только необходимые функции. В случае, если известно заранее, что все функции из библиотеки будут использоваться, то неявное подключение более удобно и позволяет автоматически подключать все функции при запуске программы.

Таким образом, выбор метода подключения динамических библиотек зависит от конкретных требований проекта и необходимости использования определенных функций.