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 №5

«Изучение работы DLL»

Отчёт по лабораторной работе дисциплины

«Технологии программирования»

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

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

Киров 2022

**Задание:** Создать на основе лабораторной работы 3 библиотечные модули и разработать новое приложение. Интерфейсную часть приложения оформить в виде библиотеки неявной загрузки, а основной функционал - библиотеки явной загрузки. Проанализировать особенности вызовов и функционирование библиотек.

**Экранные формы:**



Рисунок 1 – Результат умножения.



Рисунок 2 – результат деления.

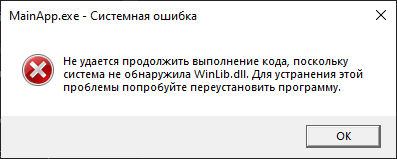


Рисунок 3 – Результат без DLL неявного связывания.

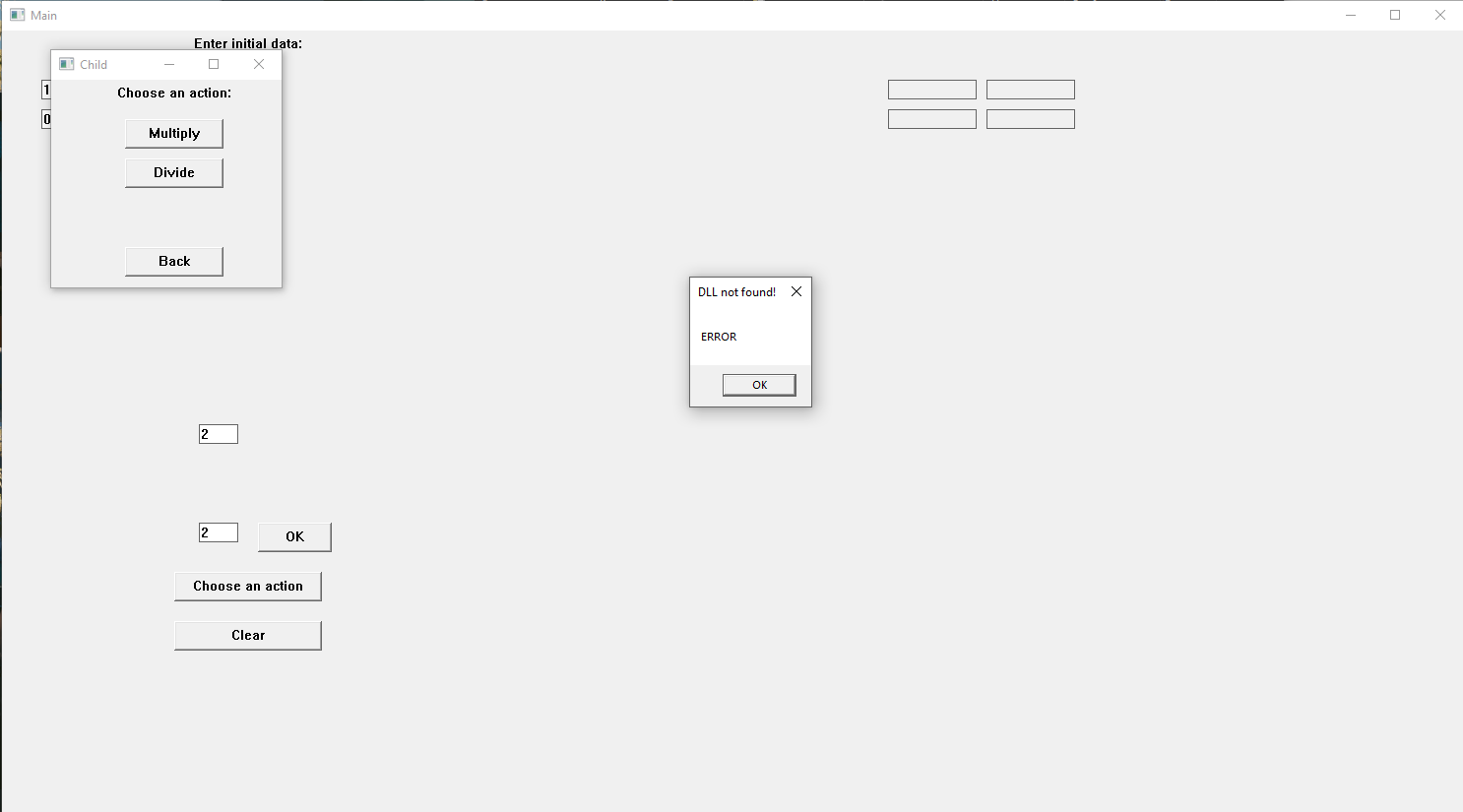


Рисунок 4 – Результат без DLL явного связывания.

**Листинг:**

WinLib.h:

#pragma once

#include <Windows.h>

#include <string>

#include <algorithm>

#include <iostream>

#include <wchar.h>

extern "C" \_\_declspec(dllexport) int Win(HINSTANCE hInstance, HINSTANCE hPrevInst, PWSTR szCmdLine, int nCmdShow);

WinLib.cpp:

#include "WinLibHeader.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

const int N = 5

//Main

HWND Static;

HWND ButtonChoose;

HWND ButtonClear;

HWND hEdit[10 \* 10];

HWND hEditSC;

HWND EditRes[10 \* 10];

HWND EditNum;

HWND ButtonOK;

HWND StaticEr;

//Child

HWND Static1;

HWND Static2;

HWND ButtonBack;

HWND ButtonDivide;

HWND ButtonMultiply;

int f = 0;

bool ff = true;

int a[10][10];

int scal = 0;

int number = 10;

int past = 0;

void Clear(HWND edit)

{

SetWindowText(edit, L"");

}

void MainWndAddWidjets(HWND hWnd)

{

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

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

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);

hEditSC = CreateWindow(L"edit", NULL, WS\_VISIBLE | WS\_CHILD | WS\_BORDER,

200, 400, 40, 20, hWnd, NULL, NULL, NULL);

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

200, 500, 40, 20, hWnd, NULL, NULL, NULL);

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

260, 500, 75, 30, hWnd, (HMENU)OnButtonOKClick, NULL, NULL);

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

200, 450, 490, 20, hWnd, NULL, NULL, NULL);

int k = 0;

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

{

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

{

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++;

}

}

}

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"Divide", WS\_CHILD | WS\_VISIBLE,

75, 80, 100, 30, hWnd, (HMENU)OnButtonDivideClick, NULL, NULL);

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

75, 40, 100, 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);

}

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 Win(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);

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:

{

HINSTANCE load = LoadLibrary(L"FuncLib.dll");

if (load == NULL)

{

MessageBox(0, L"ERROR", L"DLL not found!", MB\_OK);

}

else

{

typedef void (\*mul)(int(&mas)[10][10], int nnum);

mul multy = (mul)GetProcAddress(load, "multy");

multy(a, scal);

FreeLibrary(load);

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);

}

break;

case OnButtonDivideClick:

{

f = 2;

if (scal == 0)

{

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

break;

};

HINSTANCE load = LoadLibrary(L"FuncLib.dll");

if (load == NULL)

{

MessageBox(0, L"ERROR", L"DLL not found!", MB\_OK);

}

else

{

typedef void (\*div)(int(&mas)[10][10], int nnum);

div divide = (div)GetProcAddress(load, "divide");

divide(a, scal);

FreeLibrary(load);

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);

}

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);

}

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);

}

FuncLib.h:

#pragma once

int number = 10;

extern "C" \_\_declspec(dllexport) void multy(int (&mas)[10][10], int nnum);

extern "C" \_\_declspec(dllexport) void divide(int (&mas)[10][10], int nnum);

FuncLib.cpp:

#include "FuncLibHeader.h"

void multy(int (&mas)[10][10], int nnum)

{

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

{

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

{

mas[i][j] \*= nnum;

}

}

}

void divide(int (&mas)[10][10], int nnum)

{

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

{

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

{

mas[i][j] /= nnum;

}

}

}

MainApp.cpp (неявное связывание):

#include <Windows.h>

#include"D:\ЛЕКЦИИ\LP\2 Курс\4 семестр\Технология программирования\lab5\L5\1\WinLib\WinLibHeader.h"

extern "C" \_\_declspec(dllimport) int Win(HINSTANCE hInstance, HINSTANCE hPrevInst, PWSTR szCmdLine, int nCmdShow);

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

{

return Win(hInstance,hPrevInst, szCmdLine, nCmdShow);

}

Явное связывание:

HINSTANCE load = LoadLibrary(L"FuncLib.dll");

if (load == NULL)

{

MessageBox(0, L"ERROR", L"DLL not found!", MB\_OK);

}

else

{

typedef void (\*mul)(int(&mas)[10][10], int nnum);

mul multy = (mul)GetProcAddress(load, "multy");

multy(a, scal);

FreeLibrary(load);

...

}

**Вывод:**

В ходе выполнения лабораторной работы были полученные необходимые знания базового синтаксиса языка программирования С++. Была изучена работа с динамическими библиотеками, явное и неявное связывание.

Неявная компоновка – динамическое связывание, выполняемое во время загрузки программы, при этом код приложения ссылается на идентификаторы, содержащиеся в dll. Тем самым заставляет загрузчик неявно загружать нужную библиотеку при запуске приложения.

Явное связывание производится в момент вызова приложения, при этом предполагается, что поток явно загружает dll в адресное пространство процесса, получает виртуальный адрес необходимого dll идентификатора и вызывает функцию переменную или класс по этому адресу.

Данные знания являются фундаментальными и необходимы для дальнейшего продолжения изучения различных технологий программирования и языка С++.