МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

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

**(ФГБОУ ВПО «ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

ЛАБОРАТОРНАЯ РАБОТА № 5

«Использование библиотек динамической компоновки»

Отчёт по лабораторной работе дисциплины

«Технологии программирования»

Выполнил студент группы ИВТ-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Сычугов М.В./

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

Киров, 2018

# Задание:

Написать программу, реализующую решение СЛАУ 2 способами. Реализовать статическую и динамически подгружаемые библиотеки.

# Краткие теоретические сведения

DLL – динамическая библиотека, позволяющая многократное использование различными программными приложениями.

Формат файлов DLL придерживается тех же соглашений, что и формат исполняемых файлов, сочетая код, таблицы и ресурсы, отличаясь лишь интерпретацией некоторых полей.

Статическая компоновка имеет расширение lib, создаётся компилятором. Компоновщик извлекает необходимую функцию из библиотечного файла и интегрирует её в создаваемый исполняемый модуль.

При динамической компоновке каждый модуль загружается всего один раз, а каждая запущенная программа имеет лишь ссылку на необходимую ей функцию. Для того, чтобы некоторое приложение могло вызвать функцию, содержащуюся в DLL, образ файла DLL должен быть проецирован на адресное пространство процесса, который вызывает необходимую функцию.

Неявное связывание - это динамическое связывание во время загрузки программы, при этом код приложения ссылается на идентификаторы, содержащиеся в DLL, и тем самым заставляет загрузчик неявно подгрузить нужную библиотеку при запуске приложения. Будем считать, что исполняемый EXE модуль импортирует функции и переменные из DLL библиотеки. Тогда DLL экспортирует свои элементы в исполняемый модуль.

Явное связывание - определяет связь между требуемым DLL и клиентским приложением в процессе работы программы. Поток приложения загружает DLL в адресное пространство процесса, получает виртуальный адрес функций и вызывает эти функции по полученному адресу.

# Листинг программы

Библиотека для статического подключения

//StaticLib.h

#pragma once

double\* method\_gauss(double \*\*a, double \*b, int n);

double\* method\_iter(double\*\* a, double\* b, int n);

//StaticLib.cpp

#include "StaticLib.h"

#include <cmath>

#include <stdexcept>

#include <iostream>

#include <vector>

using namespace std;

double\* method\_gauss(double \*\*a, double \*b, int n)

{

const double eps = 0.001; // точность

double \*x = new double[n];

int k = 0;

while (k < n)

{

double max = abs(a[k][k]);

int index = k;

for (auto i = k + 1; i < n; i++)

{

if (abs(a[i][k]) > max)

{

max = abs(a[i][k]);

index = i;

}

}

if (max < eps)

{

return nullptr;

}

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

{

double temp = a[k][j];

a[k][j] = a[index][j];

a[index][j] = temp;

}

double temp = b[k];

b[k] = b[index];

b[index] = temp;

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

{

double temp = a[i][k];

if (abs(temp) < eps)

{

continue;

}

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

{

a[i][j] = a[i][j] / temp;

}

b[i] = b[i] / temp;

if (i == k)

{

continue;

}

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

{

a[i][j] = a[i][j] - a[k][j];

}

b[i] = b[i] - b[k];

}

k++;

}

for (k = n - 1; k >= 0; k--)

{

x[k] = b[k];

for (auto i = 0; i < k; i++)

{

b[i] = b[i] - a[i][k] \* x[k];

}

}

return x;

}

double\* get\_canon\_b(double\*\* a, double\* b, const int n)

{

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

{

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

{

if (i == j)

{

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

}

}

}

return b;

}

double\*\* get\_canon\_a(double\*\* a, double\* b, const int n)

{

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

{

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

{

if (i == j)

{

if (i > 0)

{

double tmp = a[i][j];

for (auto k = j; k > 0; k--)

{

a[i][k] = a[i][k - 1];

}

a[i][0] = tmp;

}

else

{

a[i][0] = a[i][j];

}

}

}

}

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

{

for (auto j = 1; j < n; j++)

{

a[i][j] = -a[i][j] / a[i][0];

}

}

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

{

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

{

if (i == j)

{

a[i][0] = a[i][0] / a[i][0];

}

}

}

return a;

}

double f\_norma(double\*\* a, const int n)

{

double max = -10, sum = 0;

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

{

for (auto j = 1; j < n; j++)

{

sum += a[i][j];

}

if (sum > max)

{

max = sum;

}

sum = 0;

}

return max;

}

double s\_norma(double\*\* a, const int n)

{

double max = -10, sum = 0;

for (auto j = 1; j < n; j++)

{

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

{

sum += a[i][j];

}

if (sum > max)

{

max = sum;

}

sum = 0;

}

return max;

}

double t\_norma(double\*\* a, const int n)

{

double max = -10, sum = 0;

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

{

for (auto j = 1; j < n; j++)

{

if (i == j)

{

sum += a[i][j] \* a[i][j];

}

}

}

return max = sqrt(sum);

}

double\* method\_iter(double\*\* a, double\* b, int n)

{

int count = 1;

//Перестановка строк

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

{

double max = a[0][j];

int numStr = j;

int numStol = j;

for (auto i = numStol; i < n; i++)

{

if (a[i][j] > max)

{

for (auto k = j; k < n; k++)

{

double tmp = a[i][k];

a[i][k] = a[numStr][k];

a[numStr][k] = tmp;

}

double tmp = b[i];

b[i] = b[numStr];

b[numStr] = tmp;

numStr = i;

max = a[i][j];

}

}

}

//-------------------

b = get\_canon\_b(a, b, n);

a = get\_canon\_a(a, b, n);

double f\_n = f\_norma(a, n);

double s\_n = s\_norma(a, n);

double t\_n = t\_norma(a, n);

if (f\_n > 1 && s\_n > 1 && t\_n > 1)

{

return nullptr;

}

double\* xk = new double[n];

double\* xn = new double[n];

vector<int> index;

index.reserve(n);

double error = 1;

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

{

xk[i] = b[i];

}

while (error > 0.001)

{

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

{

xn[i] = xk[i];

}

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

{

index.clear();

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

{

if (j != i) {

index.push\_back(j);

}

}

xk[i] = b[i];

for (auto k = 0; k < n - 1; k++)

{

xk[i] += a[i][k + 1] \* xk[index.at(k)];

}

}

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

{

if (abs(xn[i] - xk[i] <= error))

{

error = abs(xn[i] - xk[i]);

}

}

count++;

if(count > 100)

{

return nullptr;

}

}

return xk;

}

Библиотека для динамического подключения

// DynamicLib.h

#pragma once

extern "C" \_\_declspec(dllexport) double\* method\_gauss(double \*\*a, double \*b, int n);

extern "C" \_\_declspec(dllexport) double\* method\_iter(double\*\* a, double\* b, int n);

// DynamicLib.cpp:

// DinamicLib.cpp: определяет экспортированные функции для приложения DLL.

//

#include "DinamicLib.h"

Далее код такой же как и в StaticLib.cpp за исключением первой строчки

Основная программа

#include <windows.h>

#include <string>

#include <iomanip>

#include <windowsx.h>

#include <tchar.h>

#include <cmath>

//Подключение статической библиотеки

#include "StaticLib.h"

//------------------------------------------

/\*Неявное подключение динамической библиотеки

extern "C" \_\_declspec(dllexport) double\* method\_gauss(double \*\*a, double \*b, int n);

extern "C" \_\_declspec(dllexport) double\* method\_iter(double\*\* a, double\* b, int n);

\*/

/\*Явное подключение динамической библиотеки

HMODULE hDLL = NULL;

typedef double\*(\*METHODS)(double\*\*, double\*, int);

METHODS method\_gauss = NULL;

METHODS method\_iter = NULL;

\*/

using namespace std;

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

HWND hWindow;

HWND textMatr, textB, textRes, textNotRes;

HWND editRow1, editColumn1;

HWND buttonAccept, buttonShowResult;

HWND rButtonTradMed, rButtonShtrasMed;

HWND matrix\_a\_edit[10][10], matrix\_c\_edit[10];

double b[10];

int n;

int flag\_method = 0;

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

{

WNDCLASSEX wParentClass;

const int p\_x = 100, p\_y = 50, p\_nWidth = 1166, p\_nHeight = 638;

ZeroMemory(&wParentClass, sizeof(wParentClass));

wParentClass.cbSize = sizeof(wParentClass);

wParentClass.hbrBackground = (HBRUSH)COLOR\_WINDOW;

wParentClass.hInstance = hInstance;

wParentClass.lpfnWndProc = (WNDPROC)WndProc;

wParentClass.lpszClassName = L"PClass";

if (!RegisterClassEx(&wParentClass))

{

int nResult = GetLastError();

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

}

hWindow = CreateWindowEx(

NULL

L"PClass",

L"Решение СЛАУ",

WS\_OVERLAPPEDWINDOW,

p\_x,

p\_y,

p\_nWidth,

p\_nHeight,

NULL,

NULL,

hInstance,

NULL

);

if (!hWindow)

{

int nResult = GetLastError();

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

}

ShowWindow(hWindow, SW\_SHOW);

UpdateWindow(hWindow);

MSG msg;

ZeroMemory(&msg, sizeof(MSG));

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)

{

char \*err = NULL;

char \*Text = NULL;

switch (uMsg)

{

case WM\_CREATE:

{

CreateWindow(L"STATIC", L"Решение СЛАУ ", WS\_VISIBLE | WS\_CHILD,

20, 20, 250, 20, hwnd, NULL, NULL, NULL);

CreateWindow(L"STATIC", L"Количество строк и столбцов матрицы [1..9]:", WS\_VISIBLE | WS\_CHILD, 20, 50, 450, 20, hwnd, NULL, NULL, NULL);

editRow1 = CreateWindow(L"EDIT", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_NUMBER | ES\_CENTER, 333, 50, 40, 20, hwnd, NULL, NULL, NULL);

Edit\_LimitText(editRow1, 1);

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

| WS\_BORDER, 20, 80, 60, 20, hwnd, (HMENU)2, NULL, NULL);

break;

}

case WM\_COMMAND:

{

switch (LOWORD(wParam))

{

case 1:

{

ShowWindow(hwnd, SW\_HIDE);

EnableWindow(hWindow, TRUE);

SetFocus(hWindow);

break;

}

case 2:

{

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

{

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

{

DestroyWindow(matrix\_a\_edit[i][j]);

}

}

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

{

DestroyWindow(matrix\_c\_edit[i]);

}

DestroyWindow(rButtonTradMed);

DestroyWindow(rButtonShtrasMed);

DestroyWindow(buttonShowResult);

DestroyWindow(textMatr);

DestroyWindow(textB);

DestroyWindow(textNotRes);

DestroyWindow(textRes);

Text = new char[GetWindowTextLength(editRow1) + 1];

GetWindowTextA(editRow1, Text, GetWindowTextLength(editRow1) + 1);

n = atoi(Text);

delete(Text);

if (n < 1)

{

MessageBox(hwnd, L"Введите число в промежутке от 1 до 9", L"Ошибка", MB\_OK);

return -1;

}

textMatr = CreateWindow(L"STATIC", L"Матр", WS\_VISIBLE | WS\_CHILD,

20, 130, 250, 20, hwnd, NULL, NULL, NULL);

textB = CreateWindow(L"STATIC", L"Своб коэфф", WS\_VISIBLE | WS\_CHILD,

n \* 40 + 60, 130, 250, 20, hwnd, NULL, NULL, NULL);

auto i = 0;

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

{

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

{

matrix\_a\_edit[i][j] = CreateWindow(L"EDIT", L"0", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, j \* 40 + 20, 140 + i \* 25 + 25, 40, 25, hwnd, NULL, NULL, NULL);

}

}

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

{

matrix\_a\_edit[i][n] = CreateWindow(L"EDIT", L"0", WS\_VISIBLE | WS\_CHILD | WS\_BORDER, n \* 40 + 60, 140 + i \* 25 + 25, 40, 25, hwnd, NULL, NULL, NULL);

}

rButtonTradMed = CreateWindow(L"BUTTON", L"Решение СЛАУ методом Гаусса ", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | BS\_AUTORADIOBUTTON, 20, 140 + i \* 25 + 45, 300, 20, hwnd, (HMENU)4, NULL, NULL);

rButtonShtrasMed = CreateWindow(L"BUTTON", L"Решение СЛАУ методом простых итераций", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | BS\_AUTORADIOBUTTON, 20, 170 + i \* 25 + 45, 300, 20, hwnd, (HMENU)5, NULL, NULL);

buttonShowResult = CreateWindow(L"BUTTON", L"Рассчитать", WS\_VISIBLE | WS\_CHILD

| WS\_BORDER, 20, 210 + i \* 25 + 45, 150, 20, hwnd, (HMENU)6, NULL, NULL);

break;

}

case 3:

{

SendMessage(rButtonTradMed, BM\_SETCHECK, BST\_UNCHECKED, NULL);

SendMessage(rButtonShtrasMed, BM\_SETCHECK, BST\_CHECKED, NULL);

EnableWindow(buttonShowResult, TRUE);

break;

}

case 4:

{

flag\_method = 1;

EnableWindow(buttonShowResult, TRUE);

break;

}

case 5:

{

flag\_method = 2;

EnableWindow(buttonShowResult, TRUE);

break;

}

case 6:

{

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

{

DestroyWindow(matrix\_c\_edit[i]);

}

DestroyWindow(textNotRes);

double\*\* matrix\_a = new double\*[n];

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

{

matrix\_a[i] = new double[n];

}

double\* x = new double[n];

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

{

x[i] = 0;

}

try

{

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

{

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

{

Text = new char[GetWindowTextLength(matrix\_a\_edit[i][j]) + 1];

GetWindowTextA(matrix\_a\_edit[i][j],

Text, GetWindowTextLength(matrix\_a\_edit[i][j]) + 1);

matrix\_a[i][j] = strtod(Text, &err);

if (\*err)

{

throw 3;

}

delete(Text);

}

Text = new char[GetWindowTextLength(matrix\_a\_edit[i][n]) + 1];

GetWindowTextA(matrix\_a\_edit[i][n],

Text, GetWindowTextLength(matrix\_a\_edit[i][n]) + 1);

b[i] = strtod(Text, &err);

if (\*err)

{

throw 3;

}

delete(Text);

}

}

catch (...)

{

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

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

{

DestroyWindow(matrix\_c\_edit[i]);

}

DestroyWindow(textNotRes);

DestroyWindow(textRes);

return -1;

}

//При статической библиотеке и неявном подключении динамической библиотеки

if (flag\_method == 1) {

x = method\_gauss(matrix\_a, b, n);

}

else if (flag\_method == 2)

{

x = method\_iter(matrix\_a, b, n);

}

/\*Явном подключении динамической библиотеки

if (flag\_method == 1)

{

hDLL = LoadLibraryA("DinamicLib.dll");

method\_gauss = (METHODS)GetProcAddress(hDLL, "method\_gauss");

method\_iter = (METHODS)GetProcAddress(hDLL, "method\_iter");

if (hDLL == NULL)

{

MessageBox(NULL, \_T("Не найдена библиотека DLL"), \_T("Не найдена библиотека DLL"), MB\_OK);

x = nullptr;

}else

{

x = method\_gauss(matrix\_a, b, n);

FreeLibrary(hDLL);

}

}

else if (flag\_method == 2)

{

hDLL = LoadLibraryA("DinamicLib.dll");

method\_gauss = (METHODS)GetProcAddress(hDLL, "method\_gauss");

method\_iter = (METHODS)GetProcAddress(hDLL, "method\_iter");

if (hDLL == NULL)

{

MessageBox(NULL, \_T("Не найдена библиотека DLL"), \_T("Не найдена библиотека DLL"), MB\_OK);

x = nullptr;

}

else

{

x = method\_iter(matrix\_a, b, n);

FreeLibrary(hDLL);

}

}

if (x != nullptr)

{

static const double powerOfTen[] = { 1.0, 10.0, 100.0 };

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

{

const double rounded = std::floor(x[i] \* powerOfTen[2] + 0.5) / powerOfTen[2];

std::string buf = to\_string(rounded).erase(4, 10);

matrix\_c\_edit[i] = CreateWindow(L"EDIT", L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_READONLY | ES\_CENTER,

i \* 80 + 20, 240 + n \* 25 + 45, 80, 25, hwnd, NULL, NULL, NULL);

SetWindowTextA(matrix\_c\_edit[i], buf.c\_str());

}

}

else

{

textNotRes = CreateWindow(L"STATIC", L"Невозможно получить решение", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | ES\_READONLY | ES\_CENTER,

20, 240 + n \* 25 + 45, 300, 25, hwnd, NULL, NULL, NULL);

}

break;

}

}

break;

}

case WM\_DESTROY:

{

PostQuitMessage(0);

return 0;

}

}

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

}

# Экранные формы программы

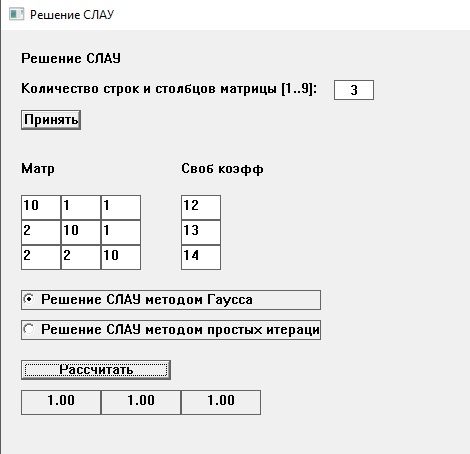


Рисунок 1 – пример работы программы

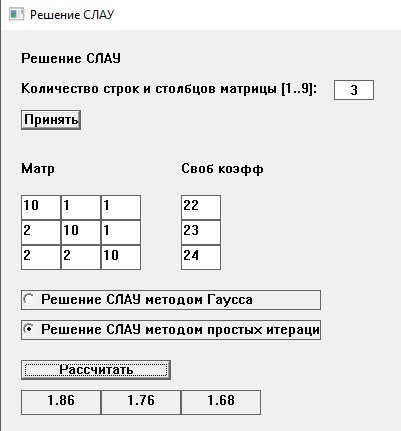


Рисунок 2 – пример работы программы

# Вывод:

В ходе выполнения лабороторной работы составлена программа,. Надождение значений было реализовано через бибилиотеки функций, которые подключаются к проекту статически и динамически.

Удобства статической компановки:

В экземпляре прикомпановоной библиотеки к исполняемому модулю и данным уже есть все, и не нужно беспокоится о том, забыли ли мы модуль или нет.

Минусы статической компаноки:

Загружаются все функции.

Большое время на загрузку статической компановки.

Удобства динамической компановки:

Загружается в память однократно, когда одно из запущенных приложений обратиться к ней, остальные приложения получают ссылку на ранее загруженную функцию.

Динамическая компановка делиться на:

Динамическое неявное связывание во время загрузки (раннее связывание).

Динамическое явное связывание во время выолнения программы (позднее связывание).

Различия динамической компановки от статической:

* Не записывается код функций в пространство приложений, используются индентификаторы тех функций, которые мы собираемся использовать.
* Библиотека будет загружаться одноременно с загрузкой исполняемого модуля. Раздел импорта возмется из заголовочных файлов, в которых написано, что мы хотим использовать и внутри нашей программы есть вызов функций, которые должны быть в этих заголовочных файлах.
* Так как в статической компановке нам не нужны DLL сразу добавили в проект библиотеку и обращаемся непосредственно указывая адреса в своем коде, что нужно вызать.

Различия неявной и явной компановки:

Неявная компановка – динамическое связывание, которое происходит на этапе загрузки программы. При этом код приложения ссылается на индетификаторы содержащиеся в динамической библиотеке тем самым застявляя загрузчик неявно загружать требую библиотеку в глобальный пул процесса.

При явной компановке требуемая DLL при работе приложения явно загружается в адресное пространство конкретного процесса и процесс получает виртуальный адрес необходимой функции и вызывает функцию по этому адресу.