ФГАОУ ВО Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа киберфизических систем и управления

**ОТЧЕТ**

**по лабораторной работе №2**

**по теме «DLL и приложение из нескольких проектов»**

по дисциплине «Практикум по программированию»

Выполнил

студент гр.23533/2 А.Д. Шурак

Проверил

ассистент В.Э. Ковалевский

Санкт-Петербург

2019

Задание

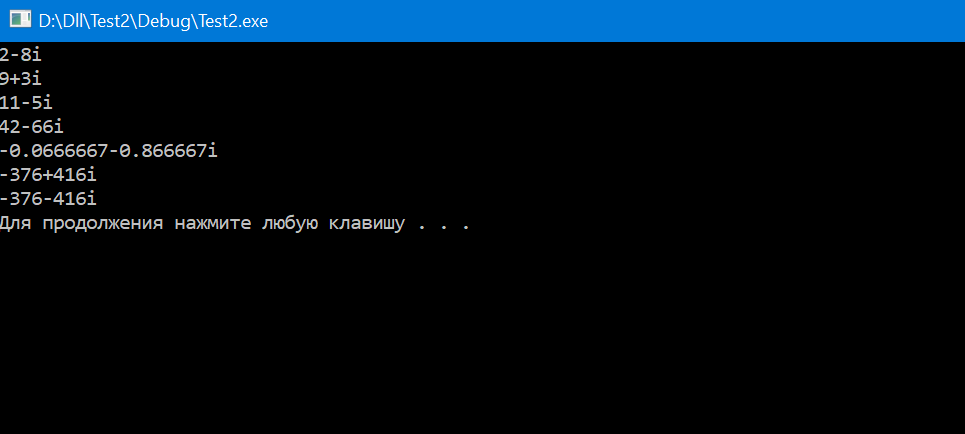
На основе лабораторной работы прошлого семестра создать динамическую библиотеку DLL и консольное приложение на языке C++, использующее эту библиотеку.

**Разработка программы**

Используя лабораторные работы из прошлого семестра, продемонстрируем работу в DLL-проекте.

1. Создаём проект DLL, который будет содержать такие файлы: d11.h, dllmain.cpp, Dll.cpp.
2. В файле Dll.cpp располагаем разработки из прошлого семестра, а именно работу с комплексными числами.
3. В файле d11.h создаём класс ComplexClass, содержащий виртуальные методы.
4. В изначально созданный проект интегрируем новый проект, содержащий такие файлы: DllClass.cpp (для загрузки библиотеки) и две копии из первого проекта. (Содержит класс Complex - производный от класса ComplexClass).
5. Мы можем использовать классы из динамической библиотеки благодаря полиморфизму, который позволяет через указатель на объект родительского класса обратиться к объекту производного класса.

**Демонстрация работы программы**

****

**Приложение**

**Проект Test2.**

#include "d11.h"

#include <iostream>

#include <windows.h>

typedef ComplexClass\* (\_\_cdecl \*ComplexClass\_factory)();

void main()

{

HINSTANCE hmodule = LoadLibrary("Dll.dll");

if (!hmodule)

{

std::cout << "Loading failed!\n";

return;

}

ComplexClass\_factory CreateClass = reinterpret\_cast<ComplexClass\_factory>(GetProcAddress(hmodule, "CreateClass"));

if (!CreateClass)

{

std::cout << "Loading failed!\n";

FreeLibrary(hmodule);

return;

}

ComplexClass\* A = CreateClass();

ComplexClass\* B = CreateClass();

ComplexClass\* C = CreateClass();

A->SetReIm(2, -8);

B->SetReIm(9, 3);

A->Out();

B->Out();

C = A->Plus(B);

C->Out();

C = A->Multiply(B);

C->Out();

C = A->Divide(B);

C->Out();

C = A->Power(3);

C->Out();

C->Maiting();

C->Out();

A->Clear();

B->Clear();

C->Clear();

FreeLibrary(hmodule);

system("pause");

}

**Проект Dll.**

#pragma once

class ComplexClass

{

public:

virtual void Clear() = 0;

virtual void Out() = 0;

virtual void Maiting() = 0;

virtual void SetReIm(float a, float b) = 0;

virtual float GetRe() = 0;

virtual float GetIm() = 0;

virtual ComplexClass\* Plus(ComplexClass\* B) = 0;

virtual ComplexClass\* Plus(float B) = 0;

virtual ComplexClass\* Minus(ComplexClass\* B) = 0;

virtual ComplexClass\* Minus(float B) = 0;

virtual ComplexClass\* Multiply(ComplexClass\* B) = 0;

virtual ComplexClass\* Multiply(float B) = 0;

virtual ComplexClass\* Divide(ComplexClass\* B) = 0;

virtual ComplexClass\* Divide(float B) = 0;

virtual ComplexClass\* Power(int Power) = 0;

};

// dllmain.cpp: определяет точку входа для приложения DLL.

#include "stdafx.h"

BOOL APIENTRY DllMain( HMODULE hModule,

DWORD ul\_reason\_for\_call,

LPVOID lpReserved

)

{

switch (ul\_reason\_for\_call)

{

case DLL\_PROCESS\_ATTACH:

case DLL\_THREAD\_ATTACH:

case DLL\_THREAD\_DETACH:

case DLL\_PROCESS\_DETACH:

break;

}

return TRUE;

}

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

//

#include "stdafx.h"

#include "d11.h"

#include <iostream>

#include <windows.h>

class Complex : public ComplexClass

{

private:

float re;

float im;

public:

Complex()

{

re = 0;

im = 0;

}

Complex(float x, float y)

{

re = x;

im = y;

}

~Complex()

{

}

void Clear()

{

delete this;

}

void Out()

{

std::cout << re;

if (im > 0) std::cout << "+";

if (im != 0) std::cout << im << "i\n";

}

void Maiting()

{

im = im \* (-1);

}

float GetRe()

{

return re;

}

float GetIm()

{

return im;

}

void SetReIm(float a, float b)

{

re = a;

im = b;

}

ComplexClass\* Plus(ComplexClass\* B)

{

ComplexClass\* C = new Complex;

C->SetReIm(GetRe() + B->GetRe(), GetIm() + B->GetIm());

return C;

}

virtual ComplexClass\* Plus(float B)

{

ComplexClass\* C = new Complex;

C->SetReIm(GetRe() + B, GetIm());

return C;

}

ComplexClass\* Minus(ComplexClass\* B)

{

ComplexClass\* C = new Complex;

C->SetReIm(GetRe() - B->GetRe(), GetIm() - B->GetIm());

return C;

}

ComplexClass\* Minus(float B)

{

ComplexClass\* C = new Complex;

C->SetReIm(GetRe() - B, GetIm());

return C;

}

ComplexClass\* Multiply(ComplexClass\* B)

{

ComplexClass\* C = new Complex;

C->SetReIm((GetRe()\*B->GetRe()) - (GetIm()\*B->GetIm()), ((GetRe()\*B->GetIm()) + (GetIm()\*B->GetRe())));

return C;

}

ComplexClass\* Multiply(float B)

{

ComplexClass\* C = new Complex;

C->SetReIm((GetRe()\*B), (GetIm()\*B));

return C;

}

ComplexClass\* Divide(ComplexClass\* B)

{

ComplexClass\* C = new Complex;

float x, y;

x = ((GetRe()\*B->GetRe() + GetIm()\*B->GetIm()) / (B->GetRe()\*B->GetRe() + B->GetIm()\*B->GetIm()));

y = ((GetIm()\*B->GetRe() - GetRe()\*B->GetIm()) / (B->GetRe()\*B->GetRe() + B->GetIm()\*B->GetIm()));

C->SetReIm(x, y);

return C;

}

ComplexClass\* Divide(float B)

{

ComplexClass\* C = new Complex;

C->SetReIm((GetRe()\*B) / (B\*B), (GetIm()\*B) / (B\*B));

return C;

}

ComplexClass\* Power(int Power)

{

ComplexClass\* C = new Complex;

C->SetReIm(GetRe(), GetIm());

if (Power > 0)

{

for (int i = 1; i < Power; i++)

{

C = C->Multiply(this);

}

}

return C;

}

};

extern "C" \_\_declspec(dllexport) ComplexClass\* \_\_cdecl CreateClass()

{

ComplexClass\* A = new Complex;

return A;

}