**Форма № Н-6.01**

ЗАТВЕРДЖЕНО

Наказ Міністерства освіти і науки

29 березня 2012 року № 384

Інститут фізико-технічних та комп’ютерних наук

Відділ комп’ютерних технологій

Кафедра програмного забезпечення комп’ютерних систем

КУРСОВИЙ ПРОЕКТ

на тему:

**Дії над комплексними числами**

Студента 2 курсу 243 групи

напряму підготовки

«Програмна інженерія»

Джурана А.С.

Керівник

Доцент кафедри МПУіК,

Лазорик В.В.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

**Члени комісії** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

Чернівці – 2015

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Інститут фізико-технічних та комп’ютерних наук

Відділ комп’ютерних технологій

Кафедра програмного забезпечення комп’ютерних систем

**ЗАТВЕРДЖУЮ**

Завідувач кафедри

доктор фіз.-мат.наук, проф.

\_\_\_\_\_\_\_\_С.Е. Остапов

”\_\_\_”\_\_\_\_\_\_\_\_\_\_2015 р.

|  |  |
| --- | --- |
| *Підпис і дата* |  |
| *Інв. № дубл.* |  |
| *Зам. інв. №* |  |
| *Підпис і дата* |  |
| *Інв. № ориг.* |  |

**Дії над комплексними числами**

**ЛИСТ ЗАТВЕРДЖЕННЯ**

**«УЗГОДЖЕНО»**

Керівник проекту

\_\_\_\_\_\_\_\_\_\_\_В.В. Лазорик

”\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_2015 р.

Виконавець

Студент 243 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.С.Джуран

”\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_ 2015 р.

Нормоконтролер

\_\_\_\_\_\_\_\_\_\_\_\_М.С. Ілащук

”\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_ 2015 р.

2015 ЛЗ

Чернівецький національний університет імені Юрія Федьковича

(назва вузу)

Інститут  *фізико-технічних та комп’ютерних наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Кафедра *\_ програмного забезпечення комп’ютерних систем\_\_\_\_\_\_\_\_*

Напрям підготовки \_ *Програмна інженерія\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**ЗАТВЕРДЖУЮ**

зав. кафедрою\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2015 р.

**ЗАВДАННЯ НА КУРСОВИЙ ПРОЕКТ**

Джурана Артема Сергійовича

(прізвище, ім’я, по батькові)

1. Тема роботи *Програма «Дії над комплексними числами»*

Затверджено наказом по університету від «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2015 р. №\_\_\_\_\_\_

1. Термін подачі студентом закінченого проекту (роботи) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Вихідні дані до проекту (роботи) *«Середовище розробки Visual Studio 2013, Гради Буч: Объектно-ориентированный анализ и проектирование»*
3. Зміст документації до кваліфікаційної роботи
4. (перелік питань, що їх належить розробити) *Розробити програму «Дії над комплексними числами»*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Перелік графічного матеріалу

(з точним зазначенням обов’язкових креслень) блок-схема (рис. 1),\_\_\_\_\_\_ Головне вікно програми (рис. 2)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Консультанти по проекту (роботі), із зазначенням розділів проекту, що стосуються їх

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Консультант | Підпис, дата | |
| Завдання видав | Завдання прийняв |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівники \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Завдання прийняв до виконання\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| №  п/п | Назва етапів кваліфікаційної роботи | Термін виконання роботи |
| 1 | Одержання технічного завдання | 01.12.2014р. |
| 2 | Аналіз літератури | 15.01.2015р. |
| 3 | Розробка інтерфейсу | 20.02.2015р. |
| 4 | Реалізація програми | 02.03.2015р. |
| 5 | Тестування та налагодження програми | 05.04.2015р. |
| 6 | Оформлення програмної документації | 14.04.2015р. |
| 7 | Представлення готової роботи | 17.04.2015р. |
| 8 | Захист роботи | згідно розкладу |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Керівник проекту\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**ЗМІСТ**

[ТЕОРЕТИЧНА ЧАСТИНА 7](#_Toc417468042)

[1. ТЕХНІЧНЕ ЗАВДАННЯ 8](#_Toc417468043)

[1.1. Підстави для розробки 8](#_Toc417468044)

[1.2. Призначення розробки 8](#_Toc417468045)

[1.3. Аналіз вимог до програмного забезпечення 8](#_Toc417468046)

[1.3.1. Функціональні вимоги 8](#_Toc417468047)

[1.3.2. Вимоги до складу та параметрів технічних засобів 8](#_Toc417468048)

[1.3.3. Вимоги до інтерфейсу 9](#_Toc417468049)

[1.3.4. Вимоги до інформаційної та програмної сумісності 9](#_Toc417468050)

[1.3.5. Вимоги до тестування програмного забезпечення 9](#_Toc417468051)

[1.4. Вимоги до програмної документації 10](#_Toc417468052)

[1.4.1. Склад супроводжувальної документації 10](#_Toc417468053)

[1.4.2. Вимоги до супроводжувальної документації 10](#_Toc417468054)

[1.5. Стадії та етапи розробки 10](#_Toc417468055)

[1.6. Порядок контролю і приймання 11](#_Toc417468056)

[ПРАКТИЧНА ЧАСТИНА 12](#_Toc417468057)

[2. АРХІТЕКТУРА, ФУНКЦІОНАЛЬНІ ТА ТЕХНІЧНІ ПОКАЗНИКИ 13](#_Toc417468058)

[2.1. Призначення та область застосування 13](#_Toc417468059)

[2.2. Опис та обґрунтування обраної архітектури 13](#_Toc417468060)

[2.3. Функціональна специфікація 14](#_Toc417468061)

[2.3.1. Опис функціональних можливостей 14](#_Toc417468062)

[2.3.2. Опис інтерфейсу користувача 14](#_Toc417468063)

[2.4. Технічна специфікація 15](#_Toc417468064)

[2.4.1. Опис і обґрунтування вхідних і вихідних даних 15](#_Toc417468065)

[2.4.2. Опис модулів 15](#_Toc417468066)

[3. КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 16](#_Toc417468067)

[3.1. Опис і обґрунтування обраних програмних засобів 16](#_Toc417468068)

[3.2. Опис програми 16](#_Toc417468069)

[3.2.1. Функціональні можливості 16](#_Toc417468070)

[3.2.2. Опис логічної структури 16](#_Toc417468071)

[3.2.3. Використані технічні засоби 17](#_Toc417468072)

[4. ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ 18](#_Toc417468073)

[4.1. Об’єкт випробувань 18](#_Toc417468074)

[4.2. Використані технічні засоби 18](#_Toc417468075)

[4.3. Порядок та методика випробувань 18](#_Toc417468076)

[4.4. Результати випробувань 19](#_Toc417468077)

[Висновок 20](#_Toc417468078)

[Список використаних джерел 21](#_Toc417468079)

[Додатки 22](#_Toc417468080)

[Class Complex 22](#_Toc417468081)

[Class Field 29](#_Toc417468082)

[Class Localize 30](#_Toc417468083)

[Class EDivisionByZero 33](#_Toc417468084)

[Class MyForm 34](#_Toc417468085)

ТЕОРЕТИЧНА ЧАСТИНА

1. ТЕХНІЧНЕ ЗАВДАННЯ

1.1. Підстави для розробки

Розробка програмного забезпечення виконується на підставі рішення засідання кафедри ПЗКС про затвердження тем дипломних та курсових робіт.

1.2. Призначення розробки

Призначенням даної розробки є створення програми для роботи з комплексними числами.

1.3. Аналіз вимог до програмного забезпечення

1.3.1. Функціональні вимоги

До програмного забезпечення висуваються наступні вимоги:

* Елементарні операції з комплексними числами
* Пошук експоненти комплексного числа
* Пошук тригонометричних і гіперболічних функцій

1.3.2. Вимоги до складу та параметрів технічних засобів

Для нормальної роботи розробленого програмного продукту необхідний персональний комп’ютер з наступними мінімальними характеристиками:

|  |  |
| --- | --- |
| Параметр | Значення |
| Процесор | Intel Core 2 Duo 1.8 Ghz |
| Пам’ять | 1 GB RAM |
| Жорсткий диск | 40 GB |

1.3.3. Вимоги до інтерфейсу

Програма повинна мати інтуїтивно зрозумілий, зручний та візуально приємний інтерфейс.

Програмне забезпечення повинно бути оформлене у вигляді стандартного вікна операційної системи Windows.

Програма повинна складатися з:

* Розділу введення значень комплексного числа
* Систему, яка включає в себе пошук елементарних функцій
* Розділу, який включає в себе пошук тригонометричних і гіперболічних функцій
* Результати обчислень

1.3.4. Вимоги до інформаційної та програмної сумісності

Необхідною вимогою до програмного забезпечення є встановлена операційна система класу Windows XP/7. Окремих вимог не висувається.

1.3.5. Вимоги до тестування програмного забезпечення

Для тестування програмного забезпечення необхідно виконати наступні дії:

* запустити програму;
* перейти до введення комплексних чисел;
* вибрати потрібну операцію із запропонованих;
* переглянути результати обрахунку;
* повторити попередні дії з іншою операцією або числом;
* закрити програму;

Після успішного виконання цих дій тестування програми може вважатися завершеним.

1.4. Вимоги до програмної документації

1.4.1. Склад супроводжувальної документації

Програмне забезпечення постачається разом із супроводжувальною документацією, в склад якої входить:

* технічне завдання;
* опис та обґрунтування вибраної архітектури;
* функціональна специфіка;
* технічна специфікація;
* опис програми;
* програма та методика випробувань;

1.4.2. Вимоги до супроводжувальної документації

Текст супроводжувальної документації набирається на комп’ютері шрифтом Times New Roman 14 пт. через 1,5 інтервал. Розміщення – книжкове.

Поля зліва – не менше 2,5 см., решта – не менше 1,5 см. Вирівнювання – по ширині. Відступ першого рядка абзацу – 1,27 см. Нумерація сторінок – наскрізна в усій роботі. Сторінки нумеруються по центру верхнього контитулу. На титульному аркуші та листі затвердження нумерація сторінок не ставиться, але вони враховуються в загальній кількості сторінок.

1.5. Стадії та етапи розробки

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів кваліфікаційної розробки | Термін виконання етапів розробки | Примітка |
| 1 | Одержання технічного завдання | 01.12.2014p. |  |
| 2 | Аналіз літератури | 15.01.2015р. |  |
| 3 | Розробка інтерфейсу | 20.02.2015р. |  |
| 4 | Реалізація програми | 02.03.2015р. |  |
| 5 | Тестування та налагодження | 05.04.2015р. |  |
| 6 | Оформлення програмної документації | 14.04.2015р. |  |
| 7 | Представлення готової роботи | 17.04.2015р. |  |
| 8 | Захист роботи | Згідно розкладу |  |

1.6. Порядок контролю і приймання

Програмне забезпечення повинне бути випробуване та протестоване на наявність помилок.

Порядок контролю і прийому проекту полягає в наступному:

* перевірка правильного функціонування програми керівником проекту;
* перевірка відповідності стандартам оформлення документації;
* перевірка правильного оформлення програмної документації керівником проекту;
* попередній захист курсової роботи;

ПРАКТИЧНА ЧАСТИНА

2. АРХІТЕКТУРА, ФУНКЦІОНАЛЬНІ ТА ТЕХНІЧНІ ПОКАЗНИКИ

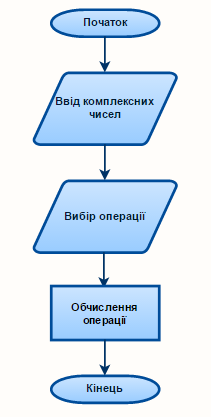
2.1. Призначення та область застосування

Призначення даної програми є операції з комплексними числами.

2.2. Опис та обґрунтування обраної архітектури

Для даного проекту була вибрана об’єктно-зорієнтована архітектура, оскільки вона є найпоширенішою та найкраще підходить для розробки програм складного рівня.

Роботу розробленої програми можна показати наступною блок-схемою, що показана на рис. 1.



*Рис. 1 Блок-схема роботи програми*

**2.3.** Функціональна специфікація

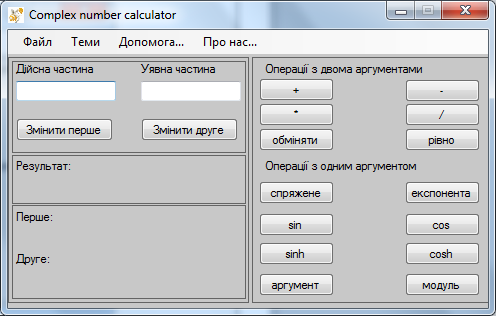
2.3.1. Опис функціональних можливостей

Розроблена програма надає можливість працювати з комплексними числами. Її функціонал дозволяє користувачеві виконувати такі дії над комплексними числами, як додавання, віднімання, множення, ділення, тригонометричні та гіперболічні операції, а також визначати експоненту.

2.**3.2.** Опис інтерфейсу користувача

Дане програмне забезпечення не вимагає особливих знань в комп’ютерній галузі, але ви не зможете правильно користуватись програмою, якщо у вас немає знань про комплексні числа. Робота з програмою потребує лише базових навичок в роботі ПК та операційною системою Windows.

Програма має лише одне вікно. З лівого боку воно містить поля для введення початкових даних і поле результату (рис. 2).



*Рис. 2. Інтерфейс програми*

**2.4.** Технічна специфікація

**2.4.1.** Опис і обґрунтування вхідних і вихідних даних

Вхідними даними для розробленого продукту є одне або два комплексних числа, які містять дійну та уявну частини. Вихіднии даними є поле з інформацію, яка показує результат обрахунку вибраної операції. Також передбачена зміна однієї мови на іншу.

**2.4.2.** Опис модулів

Робота даного програмного забезпечення реалізується в наступних модулях:

* MyForm – реалізовано головну форму програми, за допомогою якої користувач взаємодіє з програмою;
* Complex – модуль, в якому реалізовано операції роботи з комплексними числами;
* Localize – модуль, де реалізовано переклад програми на різні мови;
* Field – модуль, який використовується у класі Localize, як поле змінних;

3. КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Опис і обґрунтування обраних програмних засобів

Розробка програмного забезпечення здійснена в середовищі Visual Studio 2013 C++. Дане інтегроване середовище розробки – зручний засіб розробки віконних програм для операційної системи Windows XP/7. В ньому поєднані потужні можливості мови програмування С++ та величезний набір візуальних компонентів, які, в свою чергу, дозволяють розробнику легко швидко та якісно створювати програмні продукти з візуальним інтерфейсом. Також, візуальна бібліотека містить елементи, що спрощують діалогову взаємодію користувача та додатку (діалогові вікна, відкриття та збереження файлів, налаштування параметрів кольорів, шрифту та іншого…)

3.2. Опис програми

3.2.1. Функціональні можливості

В створеному програмному забезпеченні реалізовано наступні функції:

* Введення одного або двох комплексних чисел;
* Операції з комплексними числами;
* Перегляд результатів;

3.2.2. Опис логічної структури



*Рис 3. Діаграма прецедентів*

3.2.3. Використані технічні засоби

Розробка програмного продукту здійснювалася на персональному комп’ютері з наступною конфігурацією:

Процесор – Intel Pentium CPU B960 @ 2,2 GHz 2,2 GHz;

ОЗП – 4 GB;

Відеоадаптер – Nvidia Geforce GT 630M 1 GB;

Жорсткий диск – 500 GB;

DVD-RW – PLDS DS8A8SH ATA Device

Даний комп’ютер працює під управлінням ОС Windows 7.

4. ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

4.1. Об’єкт випробувань

Об’єктом випробувань є створене в процесі виконання курсової роботи програмне забезпечення. Для перевірки коректності роботи програми було проведено ряд тестів.

4.2. Використані технічні засоби

При проведенні тестування використовувалися технічні засоби, описані в підрозділі 3.3.1 «Використані технічні засоби». Під час проведення тестів використовувалася система ручного тестування.

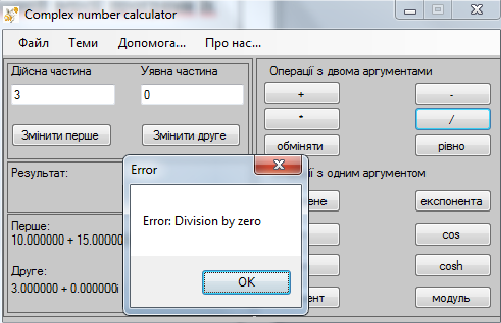
4.3. Порядок та методика випробувань

Порядок проведення випробувань програмного забезпечення:

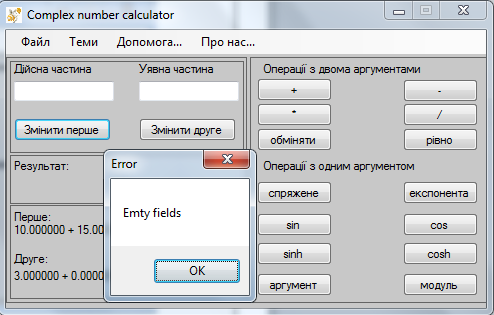
* запустити програму;
* перейти до введення комплексних чисел ;
* обрати операцію з комплексними числами;
* переглянути результати підрахунків;
* повторити попередні дії з іншою вхідними даними;
* закрити програму;

4.4. Результати випробувань

Випробування програмного продукту проводилось шляхом тестування різних вхідних даних.

Програма обробляє такі помилки, як спроба користувача поділити на 0(рис. 4) та змінити змінну з порожніми полями вводу(рис. 5). 

*Рис4. Результат випробувань 1*



*Рис5. Результат випробувань 2*

Висновок

В ході даного проекту було створено програмне забезпечення, яке дозволяє виконувати різноманітні операції з комплексними числами. Програма доволі зручна і функціональна у використанні. Тестування програмного забезпечення показало його працездатність і відповідність технічним вимогам.

Список використаних джерел

Епихин В. Е. Комплексные числа : методическая разработка для учащихся заочного отделения МММФ / В. Е. Епихин. — М.: Изд-во центра прикладных исследований при механико-математическом факультете МГУ, 2008. — 30 с.

Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++ / Гради Буч. — М.: «Издательство Бином», СПб: «Невский диалект», 1998. — 560 с.

https://ru.wikipedia.org/wiki/Одиночка\_(шаблон\_проектирования).

Додатки

Class Complex

//Complex.h

#pragma once

#include <math.h>

#include <string>

#include "EDivisionByZero.h"

using namespace std;

class Complex{

private:

//instance of exception class

EDivisionByZero eDivisionByZero;

//real and imagine parts of the complex number

double real;

double imag;

double const e = 2.7182818284;

public:

//default constructor

Complex();

//constructor that sets real part to given parametr

//imagine part becomes equal to 0

Complex(double \_real);

//constructor that takes two parametrs

Complex(double \_real, double \_imag);

//Constructor of copying

Complex(const Complex& num2);

//set-functions and get-functions are represented below

void setReal(int \_real);

void setImag(int \_imag);

double getReal();

double getImag();

//returns absolut value of the complex number

double abs();

//returns argument of the complex number

double arg();

//ads another complex number

Complex& add(const Complex& num2);

//subtracts another complex number

Complex& sub(const Complex& num2);

//multiplies on another complex number

Complex& mul(const Complex& num2);

//divides on another complex number

Complex& div(const Complex& num2);

//returns conjugated complex number

Complex& conjugate();

//returns exponent of the complex number

Complex& compExp();

//return sinus hiperbolic of the complex number

double sinh();

//return cosinus hiperbolic of the complex number

double cosh();

//returns sinus of the complex nuber

double sin();

//returns cosinus of the complex number

double cos();

//checks for equality two complex numbers

bool equals(const Complex& num2);

//returns string value of the complex number

string toString();

bool isZero();

//overloaded operators are represented below

Complex& operator +(const Complex& num);

Complex& operator +(const int num2);

Complex& operator +(const double num2);

Complex& operator -(const int num2);

Complex& operator -(const double num2);

Complex& operator -(const Complex& num2);

Complex& operator /(const int num2);

Complex& operator /(const double num2);

Complex& operator /(const Complex& num2);

Complex& operator \*(const int num2);

Complex& operator \*(const double num2);

Complex& operator \*(const Complex& num2);

bool operator ==(const int num2);

bool operator ==(const double num2);

bool operator ==(const Complex& num2);

Complex& operator +=(const int num2);

Complex& operator +=(const double num2);

Complex& operator +=(const Complex& num2);

Complex& operator -=(const int num2);

Complex& operator -=(const double num2);

Complex& operator -=(const Complex& num2);

Complex& operator \*=(const int num2);

Complex& operator \*=(const double num2);

Complex& operator \*=(const Complex& num2);

Complex& operator /=(const int num2);

Complex& operator /=(const double num2);

Complex& operator /=(const Complex& num2);

Complex& operator ++();

Complex& operator ++(int);

Complex& operator =(const Complex& num2);

};

//Complex.cpp

#include "Complex.h"

//default constructor

Complex::Complex(){

real = 0;

imag = 0;

}

//constructor that sets real part to given parametr

//imagine part becomes equal to 0

Complex::Complex(double \_real) :Complex()

{

real = \_real;

}

//constructor that takes two parametrs

Complex::Complex(double \_real, double \_imag) :Complex()

{

real = \_real;

imag = \_imag;

}

//Constructor of copying

Complex::Complex(const Complex& num2){

real = num2.real;

imag = num2.imag;

}

//set-functions and get-functions are represented below

void Complex::setReal(int \_real){

real = \_real;

}

void Complex::setImag(int \_imag){

imag = \_imag;

}

double Complex::getReal(){

return real;

}

double Complex::getImag(){

return imag;

}

//returns absolut value of the complex number

double Complex::abs()

{

//return sqrt(real\*real + imag\*imag);

double res;

try{

res = sqrt(real\*real + imag\*imag);

}

catch (...){

exit(1);

}

return res;

}

//returns argument of the complex number

double Complex::arg()

{

//return atan(imag / real);

double res;

try{

res = atan(imag / real);

}

catch (...){

exit(1);

}

return res;

}

//ads another complex number

Complex& Complex::add(const Complex& num2)

{

return Complex(real + num2.real, imag + num2.imag);

}

//subtracts another complex number

Complex& Complex::sub(const Complex& num2)

{

return Complex(real - num2.real, imag - num2.imag);

}

//multiplies on another complex number

Complex& Complex::mul(const Complex& num2)

{

double tmpReal = real\*num2.real - imag\*num2.imag;

double tmpImag = real\*num2.imag + imag\*num2.real;

return Complex(tmpReal, tmpImag);

}

//divides on another complex number

Complex& Complex::div(const Complex& num2)

{

if (num2.real == 0 || num2.imag == 0)

throw eDivisionByZero;

double tmpReal = (real\*num2.real + imag\*num2.imag) / (pow(num2.real, 2) + pow(num2.imag, 2));

double tmpImag = (imag\*num2.real - real\*num2.imag) / (pow(num2.real, 2) + pow(num2.imag, 2));

return Complex(tmpReal, tmpImag);

}

//returns conjugated complex number

Complex& Complex::conjugate()

{

return Complex(real\*real - imag\*imag, 0);

}

//returns exponent of the complex number

Complex& Complex::compExp(){

return Complex(pow(exp(1), real), pow(exp(1), imag));

//return Complex(pow(exp(), real), pow(exp(), imagine));

}

double Complex::sinh()

{

return (pow(exp(1), arg()) - pow(exp(1), arg()) / 2);

}

double Complex::cosh()

{

return (pow(exp(1), arg()) + pow(exp(1), arg()) / 2);

}

double Complex::sin()

{

return real / abs();

}

double Complex::cos()

{

return imag / abs();

}

//checks for equality two complex numbers

bool Complex::equals(const Complex& num2){

return (real == num2.real && imag == num2.imag);

}

//returns string value of the complex number

string Complex::toString()

{

if (imag < 0) {

return to\_string(real) + " " + to\_string(imag) + "i";

}

else

{

return to\_string(real) + " + " + to\_string(imag) + "i";

}

return "Unexpected error in converting to string";

}

bool Complex::isZero()

{

return (real == 0 || imag == 0);

}

//overloaded operators are represented below

Complex& Complex::operator +(const Complex& num){

return this->add(num);

}

Complex& Complex::operator +(const int num2){

return Complex(real + num2, imag);

}

Complex& Complex::operator +(const double num2){

return Complex(real + num2, imag);

}

Complex& Complex::operator -(const int num2){

return Complex(real - num2, imag);

}

Complex& Complex::operator -(const double num2){

return Complex(real - num2, imag);

}

Complex& Complex::operator -(const Complex& num2){

return this->sub(num2);

}

Complex& Complex::operator /(const int num2){

double tmpReal = real\*num2 / pow(num2, 2);

double tmpImag = imag\*num2 / pow(num2, 2);

return Complex(tmpReal, tmpImag);

}

Complex& Complex::operator /(const double num2){

double tmpReal = real\*num2 / pow(num2, 2);

double tmpImag = imag\*num2 / pow(num2, 2);

return Complex(tmpReal, tmpImag);

}

Complex& Complex::operator /(const Complex& num2){

return this->div(num2);

}

Complex& Complex::operator \*(const int num2){

double tmpReal = real\*num2;

double tmpImag = imag\*num2;

return Complex(tmpReal, tmpImag);

}

Complex& Complex::operator \*(const double num2){

double tmpReal = real\*num2;

double tmpImag = imag\*num2;

return Complex(tmpReal, tmpImag);

}

Complex& Complex::operator \*(const Complex& num2){

return this->mul(num2);

}

bool Complex::operator ==(const int num2){

return (real == num2 && imag == 0);

}

bool Complex::operator ==(const double num2){

return (real == num2 && imag == 0);

}

bool Complex::operator ==(const Complex& num2){

return this->equals(num2);

}

Complex& Complex::operator +=(const int num2){

real += num2;

return \*this;

}

Complex& Complex::operator +=(const double num2){

real += num2;

return \*this;

}

Complex& Complex::operator +=(const Complex& num2){

real += num2.real;

imag += num2.imag;

return \*this;

}

Complex& Complex::operator -=(const int num2){

real -= num2;

return \*this;

}

Complex& Complex::operator -=(const double num2){

real -= num2;

return \*this;

}

Complex& Complex::operator -=(const Complex& num2){

double tmpReal = real - num2.real;

double tmpImag = imag - num2.imag;

real = tmpReal;

imag = tmpImag;

return \*this;

}

Complex& Complex::operator \*=(const int num2){

double tmpReal = real\*num2;

double tmpImag = imag\*num2;

real = tmpReal;

imag = tmpImag;

return \*this;

}

Complex& Complex::operator \*=(const double num2){

double tmpReal = real\*num2;

double tmpImag = imag\*num2;

real = tmpReal;

imag = tmpImag;

return \*this;

}

Complex& Complex::operator \*=(const Complex& num2){

double tmpReal = real\*num2.real - imag\*num2.imag;

double tmpImag = real\*num2.imag + imag\*num2.real;

real = tmpReal;

imag = tmpImag;

return \*this;

}

Complex& Complex::operator /=(const int num2){

double tmpReal = real\*num2 / pow(num2, 2);

double tmpImag = imag\*num2 / pow(num2, 2);

real = tmpReal;

imag = tmpImag;

return \*this;

}

Complex& Complex::operator /=(const double num2){

double tmpReal = real\*num2 / pow(num2, 2);

double tmpImag = imag\*num2 / pow(num2, 2);

real = tmpReal;

imag = tmpImag;

return \*this;

}

Complex& Complex::operator /=(const Complex& num2){

double tmpReal = (real\*num2.real + imag\*num2.imag) / (pow(num2.real, 2) + pow(num2.imag, 2));

double tmpImag = (imag\*num2.real - real\*num2.imag) / (pow(num2.real, 2) + pow(num2.imag, 2));

real = tmpReal;

imag = tmpImag;

return \*this;

}

Complex& Complex::operator ++(){

real++;

imag++;

return \*this;

}

Complex& Complex::operator ++(int){

real++;

imag++;

return \*this;

}

Complex& Complex::operator =(const Complex& num2){

real = num2.real;

imag = num2.imag;

return \*this;

}

Class Field

//Field.h

#pragma once

#include <string>

using namespace std;

class Field

{

private:

string \_value;

public:

System::String^ get();

void set(string value);

};

//Field.cpp

#include "Field.h"

System::String^ Field::get(){

return gcnew System::String(\_value.c\_str());

}

void Field::set(string value){

\_value = value;

}

Class Localize

//Localize.h

#pragma once

#include <string>

//includes iostream to enhance getline() function

#include <iostream>

#include <fstream>

#include <vector>

#include "Field.h"

using namespace std;

class Localize

{

private:

static Localize instance;

Localize(bool cond);

Localize();

public:

static Localize getInstance();

private:

vector< string > \_cultures;

string \_culture;

const int NonePos = 13;

public:

//Buttons

Field chngFirst;

Field chngSecond;

Field equals;

Field arg;

Field exp;

Field conj;

Field abs;

Field swap;

//Labels

Field real;

Field imagine;

Field result;

Field first;

Field second;

Field none;

Field twoOperandOperations;

Field oneOperandOperations;

//Menu strip

Field file;

Field language;

Field exit;

Field themes;

Field white;

Field blue;

Field green;

Field grey;

Field coral;

Field salmon;

Field help;

Field about;

private:

//This functions must be private

//Direct acces is not allowed

void changeFields(string fileAdrees);

//is used in the changeFields private function

//Returns string line from the file

string getValue(ifstream& file);

bool cultureExists(string culture);

//gets function pointer. Refreshes all the text on the form

static void refresh();

public:

void setCulture(string culture);

};

//Localize.cpp

#include "Localize.h"

Localize Localize::instance;

Localize::Localize(){

\_cultures = { "Ua", "Eng" };

}

Localize::Localize(bool cond)

{

Localize instance = this;

}

Localize Localize::getInstance()

{

\_\_if\_not\_exists(instance)

{

instance = Localize(true);

}

return instance;

}

//This functions must be private

//Direct acces is not allowed

void Localize::changeFields(string fileAdrees){

string value;

ifstream cultureFile(fileAdrees);

if (cultureFile){

chngFirst.set(getValue(cultureFile));

chngSecond.set(getValue(cultureFile));

equals.set(getValue(cultureFile));

arg.set(getValue(cultureFile));

exp.set(getValue(cultureFile));

conj.set(getValue(cultureFile));

abs.set(getValue(cultureFile));

swap.set(getValue(cultureFile));

real.set(getValue(cultureFile));

imagine.set(getValue(cultureFile));

result.set(getValue(cultureFile));

first.set(getValue(cultureFile));

second.set(getValue(cultureFile));

//none.set(getValue(cultureFile));

twoOperandOperations.set(getValue(cultureFile));

oneOperandOperations.set(getValue(cultureFile));

file.set(getValue(cultureFile));

language.set(getValue(cultureFile));

exit.set(getValue(cultureFile));

themes.set(getValue(cultureFile));

white.set(getValue(cultureFile));

blue.set(getValue(cultureFile));

green.set(getValue(cultureFile));

grey.set(getValue(cultureFile));

coral.set(getValue(cultureFile));

salmon.set(getValue(cultureFile));

help.set(getValue(cultureFile));

about.set(getValue(cultureFile));

cultureFile.close();

}

}

//is used in the changeFields private function

//Returns string line from the file

string Localize::getValue(ifstream& file){

string value;

getline(file, value);

return value;

}

bool Localize::cultureExists(string culture){

for (unsigned int i = 0; i < \_cultures.size(); i++){

if (culture == \_cultures[i])

return true;

}

return false;

}

void Localize::setCulture(string culture){

if (cultureExists(culture)){

\_culture = culture;

string filePath = culture + ".clt";

changeFields(filePath);

//refresh();

}

}

Class EDivisionByZero

//EDivisionByZero.h

#pragma once

#include <exception>

class EDivisionByZero

{

public:

EDivisionByZero();

System::String^ EDivisionByZero::toString();

};

//EDivisionByZero.cpp

#include "EDivisionByZero.h"

EDivisionByZero::EDivisionByZero()

{

}

System::String^ EDivisionByZero::toString()

{

return gcnew System::String("Error: Division by zero");

}

Class MyForm

//MyForm.h

#pragma once

#include <fstream>

#include "Complex.h"

#include "Localize.h"

#include <msclr\marshal\_cppstd.h>

using namespace System::Runtime::InteropServices;

string toString(System::String^ value){

return msclr::interop::marshal\_as<std::string>(value);

}

System::String^ toSysString(string \_value){

return gcnew System::String(\_value.c\_str());

}

double SysStringToDouble(System::String^ stringValue){

/\*double value;

try

{\*/

double value = double::Parse(stringValue);

/\*}

catch (System::FormatException ^e)

{

return 0;

}

catch (...)

{

return 0;

}\*/

return value;

}

namespace CourseWork {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Main form of the Project

/// </summary>

//Global variables

Complex op1;

Complex op2;

bool op1Put = false;

bool op2Put = false;

double res;

Localize localize = Localize::getInstance();

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//Initializies form

//

}

void showMess(string str){

MessageBox::Show(toSysString(str));

}

void refresh()

{

//Buttons

bChangeFirst->Text = localize.chngFirst.get();

bChangeSecond->Text = localize.chngSecond.get();

bEquals->Text = localize.equals.get();

bArg->Text = localize.arg.get();

bExp->Text = localize.exp.get();

bConj->Text = localize.conj.get();

bAbs->Text = localize.abs.get();

bSwap->Text = localize.swap.get();

//Labels

lReal->Text = localize.real.get();

lImagine->Text = localize.imagine.get();

lFirst->Text = localize.first.get();

lSecond->Text = localize.second.get();

//if (localize.isNone(toString(lFirstOutput->Text)));

/\*if (lFirstOutput->Text == "None")

lFirstOutput->Text = localize.none.get();

//if (lSecondOutput->Text == "None")

if (localize.isNone(toString(lSecondOutput->Text)))

lSecondOutput->Text = localize.none.get();\*/

lResult->Text = localize.result.get();

lBothArg->Text = localize.twoOperandOperations.get();

lOneArg->Text = localize.oneOperandOperations.get();

//Menu strip

fileToolStripMenuItem->Text = localize.file.get();

languageToolStripMenuItem->Text = localize.language.get();

exitToolStripMenuItem->Text = localize.exit.get();

themesToolStripMenuItem->Text = localize.themes.get();

whiteToolStripMenuItem->Text = localize.white.get();

blueToolStripMenuItem->Text = localize.blue.get();

greenToolStripMenuItem->Text = localize.green.get();

greyToolStripMenuItem->Text = localize.grey.get();

coralToolStripMenuItem->Text = localize.coral.get();

salmonToolStripMenuItem->Text = localize.salmon.get();

helpToolStripMenuItem->Text = localize.help.get();

aboutToolStripMenuItem->Text = localize.about.get();

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

//Text Boxes

private: System::Windows::Forms::TextBox^ tReal;

private: System::Windows::Forms::TextBox^ tImagine;

//Labels

private: System::Windows::Forms::Label^ lReal;

private: System::Windows::Forms::Label^ lImagine;

private: System::Windows::Forms::Label^ lSecondOutput;

private: System::Windows::Forms::Label^ lFirstOutput;

private: System::Windows::Forms::Label^ lFirst;

private: System::Windows::Forms::Label^ lSecond;

private: System::Windows::Forms::Label^ lOneArg;

private: System::Windows::Forms::Label^ lBothArg;

private: System::Windows::Forms::Label^ lResultOutput;

private: System::Windows::Forms::Label^ lResult;

//Buttons

private: System::Windows::Forms::Button^ bChangeFirst;

private: System::Windows::Forms::Button^ bChangeSecond;

private: System::Windows::Forms::Button^ bEquals;

private: System::Windows::Forms::Button^ bCos;

private: System::Windows::Forms::Button^ bSin;

private: System::Windows::Forms::Button^ bDiv;

private: System::Windows::Forms::Button^ bMul;

private: System::Windows::Forms::Button^ bSub;

private: System::Windows::Forms::Button^ bAdd;

private: System::Windows::Forms::Button^ bAbs;

private: System::Windows::Forms::Button^ bArg;

private: System::Windows::Forms::Button^ bConj;

private: System::Windows::Forms::Button^ bExp;

private: System::Windows::Forms::Button^ bSwap;

private: System::Windows::Forms::Button^ bCosh;

private: System::Windows::Forms::Button^ bSinh;

//Panels

private: System::Windows::Forms::Panel^ pNumbers;

private: System::Windows::Forms::Panel^ pResult;

private: System::Windows::Forms::Panel^ pOutputNumber;

private: System::Windows::Forms::Panel^ pChangeNumber;

private: System::Windows::Forms::Panel^ pOperations;

//Menu strip toolbar

private: System::Windows::Forms::MenuStrip^ menuStrip;

private: System::Windows::Forms::ToolStripMenuItem^ fileToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ languageToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ englishToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ ukrainianToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ exitToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ helpToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ aboutToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ themesToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ whiteToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ blueToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ greyToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ coralToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ salmonToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ greenToolStripMenuItem;

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MyForm::typeid));

this->tReal = (gcnew System::Windows::Forms::TextBox());

this->tImagine = (gcnew System::Windows::Forms::TextBox());

this->lReal = (gcnew System::Windows::Forms::Label());

this->lImagine = (gcnew System::Windows::Forms::Label());

this->bChangeFirst = (gcnew System::Windows::Forms::Button());

this->bChangeSecond = (gcnew System::Windows::Forms::Button());

this->pNumbers = (gcnew System::Windows::Forms::Panel());

this->pResult = (gcnew System::Windows::Forms::Panel());

this->lResult = (gcnew System::Windows::Forms::Label());

this->lResultOutput = (gcnew System::Windows::Forms::Label());

this->pChangeNumber = (gcnew System::Windows::Forms::Panel());

this->pOutputNumber = (gcnew System::Windows::Forms::Panel());

this->lSecondOutput = (gcnew System::Windows::Forms::Label());

this->lFirstOutput = (gcnew System::Windows::Forms::Label());

this->lFirst = (gcnew System::Windows::Forms::Label());

this->lSecond = (gcnew System::Windows::Forms::Label());

this->pOperations = (gcnew System::Windows::Forms::Panel());

this->bCosh = (gcnew System::Windows::Forms::Button());

this->bSinh = (gcnew System::Windows::Forms::Button());

this->bSwap = (gcnew System::Windows::Forms::Button());

this->lOneArg = (gcnew System::Windows::Forms::Label());

this->lBothArg = (gcnew System::Windows::Forms::Label());

this->bConj = (gcnew System::Windows::Forms::Button());

this->bAbs = (gcnew System::Windows::Forms::Button());

this->bArg = (gcnew System::Windows::Forms::Button());

this->bExp = (gcnew System::Windows::Forms::Button());

this->bEquals = (gcnew System::Windows::Forms::Button());

this->bCos = (gcnew System::Windows::Forms::Button());

this->bSin = (gcnew System::Windows::Forms::Button());

this->bDiv = (gcnew System::Windows::Forms::Button());

this->bMul = (gcnew System::Windows::Forms::Button());

this->bSub = (gcnew System::Windows::Forms::Button());

this->bAdd = (gcnew System::Windows::Forms::Button());

this->menuStrip = (gcnew System::Windows::Forms::MenuStrip());

this->fileToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->languageToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->englishToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->ukrainianToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->exitToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->themesToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->whiteToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->blueToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->greyToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->coralToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->salmonToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->greenToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->helpToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->aboutToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->pNumbers->SuspendLayout();

this->pResult->SuspendLayout();

this->pChangeNumber->SuspendLayout();

this->pOutputNumber->SuspendLayout();

this->pOperations->SuspendLayout();

this->menuStrip->SuspendLayout();

this->SuspendLayout();

//

// tReal

//

this->tReal->Location = System::Drawing::Point(3, 19);

this->tReal->Name = L"tReal";

this->tReal->Size = System::Drawing::Size(100, 20);

this->tReal->TabIndex = 0;

//

// tImagine

//

this->tImagine->Location = System::Drawing::Point(128, 19);

this->tImagine->Name = L"tImagine";

this->tImagine->Size = System::Drawing::Size(100, 20);

this->tImagine->TabIndex = 1;

//

// lReal

//

this->lReal->AutoSize = true;

this->lReal->Location = System::Drawing::Point(0, 0);

this->lReal->Name = L"lReal";

this->lReal->Size = System::Drawing::Size(29, 13);

this->lReal->TabIndex = 2;

this->lReal->Text = L"Real";

//

// lImagine

//

this->lImagine->AutoSize = true;

this->lImagine->Location = System::Drawing::Point(125, 0);

this->lImagine->Name = L"lImagine";

this->lImagine->Size = System::Drawing::Size(44, 13);

this->lImagine->TabIndex = 3;

this->lImagine->Text = L"Imagine";

//

// bChangeFirst

//

this->bChangeFirst->Location = System::Drawing::Point(3, 56);

this->bChangeFirst->Name = L"bChangeFirst";

this->bChangeFirst->Size = System::Drawing::Size(97, 23);

this->bChangeFirst->TabIndex = 2;

this->bChangeFirst->Text = L"Change first";

this->bChangeFirst->UseVisualStyleBackColor = true;

this->bChangeFirst->Click += gcnew System::EventHandler(this, &MyForm::bChangeFirst\_Click);

//

// bChangeSecond

//

this->bChangeSecond->Location = System::Drawing::Point(128, 56);

this->bChangeSecond->Name = L"bChangeSecond";

this->bChangeSecond->Size = System::Drawing::Size(97, 23);

this->bChangeSecond->TabIndex = 3;

this->bChangeSecond->Text = L"Change second";

this->bChangeSecond->UseVisualStyleBackColor = true;

this->bChangeSecond->Click += gcnew System::EventHandler(this, &MyForm::bChangeSecond\_Click);

//

// pNumbers

//

this->pNumbers->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->pNumbers->Controls->Add(this->pResult);

this->pNumbers->Controls->Add(this->pChangeNumber);

this->pNumbers->Controls->Add(this->pOutputNumber);

this->pNumbers->Location = System::Drawing::Point(0, 27);

this->pNumbers->Name = L"pNumbers";

this->pNumbers->Size = System::Drawing::Size(241, 246);

this->pNumbers->TabIndex = 6;

//

// pResult

//

this->pResult->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->pResult->Controls->Add(this->lResult);

this->pResult->Controls->Add(this->lResultOutput);

this->pResult->Location = System::Drawing::Point(3, 97);

this->pResult->Name = L"pResult";

this->pResult->Size = System::Drawing::Size(234, 49);

this->pResult->TabIndex = 3;

//

// lResult

//

this->lResult->AutoSize = true;

this->lResult->Location = System::Drawing::Point(0, 3);

this->lResult->Name = L"lResult";

this->lResult->Size = System::Drawing::Size(40, 13);

this->lResult->TabIndex = 1;

this->lResult->Text = L"Result:";

//

// lResultOutput

//

this->lResultOutput->AutoSize = true;

this->lResultOutput->Location = System::Drawing::Point(3, 16);

this->lResultOutput->Name = L"lResultOutput";

this->lResultOutput->Size = System::Drawing::Size(0, 13);

this->lResultOutput->TabIndex = 2;

//

// pChangeNumber

//

this->pChangeNumber->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->pChangeNumber->Controls->Add(this->lReal);

this->pChangeNumber->Controls->Add(this->tReal);

this->pChangeNumber->Controls->Add(this->lImagine);

this->pChangeNumber->Controls->Add(this->bChangeSecond);

this->pChangeNumber->Controls->Add(this->bChangeFirst);

this->pChangeNumber->Controls->Add(this->tImagine);

this->pChangeNumber->Location = System::Drawing::Point(3, 3);

this->pChangeNumber->Name = L"pChangeNumber";

this->pChangeNumber->Size = System::Drawing::Size(234, 92);

this->pChangeNumber->TabIndex = 0;

//

// pOutputNumber

//

this->pOutputNumber->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->pOutputNumber->Controls->Add(this->lSecondOutput);

this->pOutputNumber->Controls->Add(this->lFirstOutput);

this->pOutputNumber->Controls->Add(this->lFirst);

this->pOutputNumber->Controls->Add(this->lSecond);

this->pOutputNumber->Location = System::Drawing::Point(3, 147);

this->pOutputNumber->Name = L"pOutputNumber";

this->pOutputNumber->Size = System::Drawing::Size(234, 94);

this->pOutputNumber->TabIndex = 0;

//

// lSecondOutput

//

this->lSecondOutput->AutoSize = true;

this->lSecondOutput->Location = System::Drawing::Point(0, 63);

this->lSecondOutput->Name = L"lSecondOutput";

this->lSecondOutput->Size = System::Drawing::Size(0, 13);

this->lSecondOutput->TabIndex = 9;

//

// lFirstOutput

//

this->lFirstOutput->AutoSize = true;

this->lFirstOutput->Location = System::Drawing::Point(0, 17);

this->lFirstOutput->Name = L"lFirstOutput";

this->lFirstOutput->Size = System::Drawing::Size(0, 13);

this->lFirstOutput->TabIndex = 8;

//

// lFirst

//

this->lFirst->AutoSize = true;

this->lFirst->Location = System::Drawing::Point(0, 4);

this->lFirst->Name = L"lFirst";

this->lFirst->Size = System::Drawing::Size(29, 13);

this->lFirst->TabIndex = 6;

this->lFirst->Text = L"First:";

//

// lSecond

//

this->lSecond->AutoSize = true;

this->lSecond->Location = System::Drawing::Point(0, 46);

this->lSecond->Name = L"lSecond";

this->lSecond->Size = System::Drawing::Size(47, 13);

this->lSecond->TabIndex = 7;

this->lSecond->Text = L"Second:";

//

// pOperations

//

this->pOperations->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->pOperations->Controls->Add(this->bCosh);

this->pOperations->Controls->Add(this->bSinh);

this->pOperations->Controls->Add(this->bSwap);

this->pOperations->Controls->Add(this->lOneArg);

this->pOperations->Controls->Add(this->lBothArg);

this->pOperations->Controls->Add(this->bConj);

this->pOperations->Controls->Add(this->bAbs);

this->pOperations->Controls->Add(this->bArg);

this->pOperations->Controls->Add(this->bExp);

this->pOperations->Controls->Add(this->bEquals);

this->pOperations->Controls->Add(this->bCos);

this->pOperations->Controls->Add(this->bSin);

this->pOperations->Controls->Add(this->bDiv);

this->pOperations->Controls->Add(this->bMul);

this->pOperations->Controls->Add(this->bSub);

this->pOperations->Controls->Add(this->bAdd);

this->pOperations->Location = System::Drawing::Point(244, 27);

this->pOperations->Name = L"pOperations";

this->pOperations->Size = System::Drawing::Size(241, 246);

this->pOperations->TabIndex = 7;

//

// bCosh

//

this->bCosh->Location = System::Drawing::Point(152, 184);

this->bCosh->Name = L"bCosh";

this->bCosh->Size = System::Drawing::Size(75, 23);

this->bCosh->TabIndex = 15;

this->bCosh->Text = L"cosh";

this->bCosh->UseVisualStyleBackColor = true;

this->bCosh->Click += gcnew System::EventHandler(this, &MyForm::bCosh\_Click);

//

// bSinh

//

this->bSinh->Location = System::Drawing::Point(6, 184);

this->bSinh->Name = L"bSinh";

this->bSinh->Size = System::Drawing::Size(75, 23);

this->bSinh->TabIndex = 14;

this->bSinh->Text = L"sinh";

this->bSinh->UseVisualStyleBackColor = true;

this->bSinh->Click += gcnew System::EventHandler(this, &MyForm::bSinh\_Click);

//

// bSwap

//

this->bSwap->Location = System::Drawing::Point(6, 70);

this->bSwap->Name = L"bSwap";

this->bSwap->Size = System::Drawing::Size(75, 23);

this->bSwap->TabIndex = 8;

this->bSwap->Text = L"swap";

this->bSwap->UseVisualStyleBackColor = true;

this->bSwap->Click += gcnew System::EventHandler(this, &MyForm::bSwap\_Click);

//

// lOneArg

//

this->lOneArg->AutoSize = true;

this->lOneArg->Location = System::Drawing::Point(10, 101);

this->lOneArg->Name = L"lOneArg";

this->lOneArg->Size = System::Drawing::Size(154, 13);

this->lOneArg->TabIndex = 12;

this->lOneArg->Text = L"Operations with first arguments:";

//

// lBothArg

//

this->lBothArg->AutoSize = true;

this->lBothArg->Location = System::Drawing::Point(10, 4);

this->lBothArg->Name = L"lBothArg";

this->lBothArg->Size = System::Drawing::Size(159, 13);

this->lBothArg->TabIndex = 11;

this->lBothArg->Text = L"Operations with both arguments:";

//

// bConj

//

this->bConj->Location = System::Drawing::Point(6, 123);

this->bConj->Name = L"bConj";

this->bConj->Size = System::Drawing::Size(75, 23);

this->bConj->TabIndex = 10;

this->bConj->Text = L"Conj";

this->bConj->UseVisualStyleBackColor = true;

this->bConj->Click += gcnew System::EventHandler(this, &MyForm::bConj\_Click);

//

// bAbs

//

this->bAbs->Location = System::Drawing::Point(152, 215);

this->bAbs->Name = L"bAbs";

this->bAbs->Size = System::Drawing::Size(75, 23);

this->bAbs->TabIndex = 17;

this->bAbs->Text = L"abs";

this->bAbs->UseVisualStyleBackColor = true;

this->bAbs->Click += gcnew System::EventHandler(this, &MyForm::bAbs\_Click);

//

// bArg

//

this->bArg->Location = System::Drawing::Point(6, 215);

this->bArg->Name = L"bArg";

this->bArg->Size = System::Drawing::Size(75, 23);

this->bArg->TabIndex = 16;

this->bArg->Text = L"arg";

this->bArg->UseVisualStyleBackColor = true;

this->bArg->Click += gcnew System::EventHandler(this, &MyForm::bArg\_Click);

//

// bExp

//

this->bExp->Location = System::Drawing::Point(152, 123);

this->bExp->Name = L"bExp";

this->bExp->Size = System::Drawing::Size(75, 23);

this->bExp->TabIndex = 11;

this->bExp->Text = L"exp";

this->bExp->UseVisualStyleBackColor = true;

this->bExp->Click += gcnew System::EventHandler(this, &MyForm::bExp\_Click);

//

// bEquals

//

this->bEquals->Location = System::Drawing::Point(152, 70);

this->bEquals->Name = L"bEquals";

this->bEquals->Size = System::Drawing::Size(75, 23);

this->bEquals->TabIndex = 9;

this->bEquals->Text = L"equals";

this->bEquals->UseVisualStyleBackColor = true;

this->bEquals->Click += gcnew System::EventHandler(this, &MyForm::bEquals\_Click);

//

// bCos

//

this->bCos->Location = System::Drawing::Point(152, 155);

this->bCos->Name = L"bCos";

this->bCos->Size = System::Drawing::Size(75, 23);

this->bCos->TabIndex = 13;

this->bCos->Text = L"cos";

this->bCos->UseVisualStyleBackColor = true;

this->bCos->Click += gcnew System::EventHandler(this, &MyForm::bCos\_Click);

//

// bSin

//

this->bSin->Location = System::Drawing::Point(6, 155);

this->bSin->Name = L"bSin";

this->bSin->Size = System::Drawing::Size(75, 23);

this->bSin->TabIndex = 12;

this->bSin->Text = L"sin";

this->bSin->UseVisualStyleBackColor = true;

this->bSin->Click += gcnew System::EventHandler(this, &MyForm::bSin\_Click);

//

// bDiv

//

this->bDiv->Location = System::Drawing::Point(152, 45);

this->bDiv->Name = L"bDiv";

this->bDiv->Size = System::Drawing::Size(75, 23);

this->bDiv->TabIndex = 7;

this->bDiv->Text = L"/";

this->bDiv->UseVisualStyleBackColor = true;

this->bDiv->Click += gcnew System::EventHandler(this, &MyForm::bDiv\_Click);

//

// bMul

//

this->bMul->Location = System::Drawing::Point(6, 45);

this->bMul->Name = L"bMul";

this->bMul->Size = System::Drawing::Size(75, 23);

this->bMul->TabIndex = 6;

this->bMul->Text = L"\*";

this->bMul->UseVisualStyleBackColor = true;

this->bMul->Click += gcnew System::EventHandler(this, &MyForm::bMul\_Click);

//

// bSub

//

this->bSub->Location = System::Drawing::Point(152, 21);

this->bSub->Name = L"bSub";

this->bSub->Size = System::Drawing::Size(75, 23);

this->bSub->TabIndex = 5;

this->bSub->Text = L"-";

this->bSub->UseVisualStyleBackColor = true;

this->bSub->Click += gcnew System::EventHandler(this, &MyForm::bSub\_Click);

//

// bAdd

//

this->bAdd->Location = System::Drawing::Point(6, 20);

this->bAdd->Name = L"bAdd";

this->bAdd->Size = System::Drawing::Size(75, 23);

this->bAdd->TabIndex = 4;

this->bAdd->Text = L"+";

this->bAdd->UseVisualStyleBackColor = true;

this->bAdd->Click += gcnew System::EventHandler(this, &MyForm::bAdd\_Click);

//

// menuStrip

//

this->menuStrip->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(4) {

this->fileToolStripMenuItem,

this->themesToolStripMenuItem, this->helpToolStripMenuItem, this->aboutToolStripMenuItem

});

this->menuStrip->Location = System::Drawing::Point(0, 0);

this->menuStrip->Name = L"menuStrip";

this->menuStrip->Size = System::Drawing::Size(480, 24);

this->menuStrip->TabIndex = 9;

this->menuStrip->Text = L"menuStrip1";

//

// fileToolStripMenuItem

//

this->fileToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {

this->languageToolStripMenuItem,

this->exitToolStripMenuItem

});

this->fileToolStripMenuItem->Name = L"fileToolStripMenuItem";

this->fileToolStripMenuItem->Size = System::Drawing::Size(37, 20);

this->fileToolStripMenuItem->Text = L"&File";

//

// languageToolStripMenuItem

//

this->languageToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {

this->englishToolStripMenuItem,

this->ukrainianToolStripMenuItem

});

this->languageToolStripMenuItem->Name = L"languageToolStripMenuItem";

this->languageToolStripMenuItem->Size = System::Drawing::Size(126, 22);

this->languageToolStripMenuItem->Text = L"Language";

//

// englishToolStripMenuItem

//

this->englishToolStripMenuItem->Name = L"englishToolStripMenuItem";

this->englishToolStripMenuItem->Size = System::Drawing::Size(134, 22);

this->englishToolStripMenuItem->Text = L"English";

this->englishToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::englishToolStripMenuItem\_Click);

//

// ukrainianToolStripMenuItem

//

this->ukrainianToolStripMenuItem->Name = L"ukrainianToolStripMenuItem";

this->ukrainianToolStripMenuItem->Size = System::Drawing::Size(134, 22);

this->ukrainianToolStripMenuItem->Text = L"Українська";

this->ukrainianToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::ukrToolStripMenuItem\_Click);

//

// exitToolStripMenuItem

//

this->exitToolStripMenuItem->Name = L"exitToolStripMenuItem";

this->exitToolStripMenuItem->Size = System::Drawing::Size(126, 22);

this->exitToolStripMenuItem->Text = L"Exit";

this->exitToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::exitToolStripMenuItem\_Click);

//

// themesToolStripMenuItem

//

this->themesToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(6) {

this->whiteToolStripMenuItem,

this->blueToolStripMenuItem, this->greyToolStripMenuItem, this->coralToolStripMenuItem, this->salmonToolStripMenuItem, this->greenToolStripMenuItem

});

this->themesToolStripMenuItem->Name = L"themesToolStripMenuItem";

this->themesToolStripMenuItem->Size = System::Drawing::Size(61, 20);

this->themesToolStripMenuItem->Text = L"Themes";

//

// whiteToolStripMenuItem

//

this->whiteToolStripMenuItem->Name = L"whiteToolStripMenuItem";

this->whiteToolStripMenuItem->Size = System::Drawing::Size(114, 22);

this->whiteToolStripMenuItem->Text = L"White";

this->whiteToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::whiteToolStripMenuItem\_Click);

//

// blueToolStripMenuItem

//

this->blueToolStripMenuItem->Name = L"blueToolStripMenuItem";

this->blueToolStripMenuItem->Size = System::Drawing::Size(114, 22);

this->blueToolStripMenuItem->Text = L"Blue";

this->blueToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::blackToolStripMenuItem\_Click);

//

// greyToolStripMenuItem

//

this->greyToolStripMenuItem->Name = L"greyToolStripMenuItem";

this->greyToolStripMenuItem->Size = System::Drawing::Size(114, 22);

this->greyToolStripMenuItem->Text = L"Grey";

this->greyToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::greyToolStripMenuItem\_Click);

//

// coralToolStripMenuItem

//

this->coralToolStripMenuItem->Name = L"coralToolStripMenuItem";

this->coralToolStripMenuItem->Size = System::Drawing::Size(114, 22);

this->coralToolStripMenuItem->Text = L"Coral";

this->coralToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::coralToolStripMenuItem\_Click);

//

// salmonToolStripMenuItem

//

this->salmonToolStripMenuItem->Name = L"salmonToolStripMenuItem";

this->salmonToolStripMenuItem->Size = System::Drawing::Size(114, 22);

this->salmonToolStripMenuItem->Text = L"Salmon";

this->salmonToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::redToolStripMenuItem\_Click);

//

// greenToolStripMenuItem

//

this->greenToolStripMenuItem->Name = L"greenToolStripMenuItem";

this->greenToolStripMenuItem->Size = System::Drawing::Size(114, 22);

this->greenToolStripMenuItem->Text = L"Green";

this->greenToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::greenToolStripMenuItem\_Click);

//

// helpToolStripMenuItem

//

this->helpToolStripMenuItem->Name = L"helpToolStripMenuItem";

this->helpToolStripMenuItem->Size = System::Drawing::Size(53, 20);

this->helpToolStripMenuItem->Text = L"Help...";

this->helpToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::helpToolStripMenuItem\_Click);

//

// aboutToolStripMenuItem

//

this->aboutToolStripMenuItem->Name = L"aboutToolStripMenuItem";

this->aboutToolStripMenuItem->Size = System::Drawing::Size(61, 20);

this->aboutToolStripMenuItem->Text = L"About...";

this->aboutToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm::aboutToolStripMenuItem\_Click);

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::SystemColors::ScrollBar;

this->ClientSize = System::Drawing::Size(480, 278);

this->Controls->Add(this->pNumbers);

this->Controls->Add(this->pOperations);

this->Controls->Add(this->menuStrip);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;

this->Icon = (cli::safe\_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));

this->MainMenuStrip = this->menuStrip;

this->MaximizeBox = false;

this->Name = L"MyForm";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"Complex number calculator";

this->Load += gcnew System::EventHandler(this, &MyForm::MyForm\_Load);

this->pNumbers->ResumeLayout(false);

this->pResult->ResumeLayout(false);

this->pResult->PerformLayout();

this->pChangeNumber->ResumeLayout(false);

this->pChangeNumber->PerformLayout();

this->pOutputNumber->ResumeLayout(false);

this->pOutputNumber->PerformLayout();

this->pOperations->ResumeLayout(false);

this->pOperations->PerformLayout();

this->menuStrip->ResumeLayout(false);

this->menuStrip->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void bChangeFirst\_Click(System::Object^ sender, System::EventArgs^ e) {

double real, imag;

if (tReal->Text == "" || tImagine->Text == "")

{

MessageBox::Show("Emty fields", "Error");

}

else

{

try

{

real = SysStringToDouble(tReal->Text);

imag = SysStringToDouble(tImagine->Text);

op1 = Complex(real, imag);

lFirstOutput->Text = toSysString(op1.toString());

}

catch (System::FormatException ^e)

{

MessageBox::Show("Error during convertion occured", "Error");

}

}

op1Put = true;

}

private: System::Void bChangeSecond\_Click(System::Object^ sender, System::EventArgs^ e) {

double real, imag;

if (tReal->Text == "" || tImagine->Text == "")

{

MessageBox::Show("Emty fields", "Error");

}

else

{

try

{

real = SysStringToDouble(tReal->Text);

imag = SysStringToDouble(tImagine->Text);

op2 = Complex(real, imag);

lSecondOutput->Text = toSysString(op2.toString());

}

catch (System::FormatException ^e)

{

MessageBox::Show("Error during convertion occured", "Error");

}

}

op2Put = true;

}

private: System::Void bAdd\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put && op2Put)

{

op1 = op1 + op2;

lResultOutput->Text = toSysString(op1.toString());

lSecondOutput->Text = toSysString(op2.toString());

lFirstOutput->Text = toSysString(op1.toString());

}

}

private: System::Void bSub\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put && op2Put)

{

op1 = op1.sub(op2);

lResultOutput->Text = toSysString(op1.toString());

lSecondOutput->Text = toSysString(op2.toString());

lFirstOutput->Text = toSysString(op1.toString());

}

}

private: System::Void bMul\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put && op2Put)

{

op1 = op1.mul(op2);

lResultOutput->Text = toSysString(op1.toString());

lSecondOutput->Text = toSysString(op2.toString());

lFirstOutput->Text = toSysString(op1.toString());

}

}

private: System::Void bDiv\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put && op2Put)

{

try

{

op1 = op1.div(op2);

lResultOutput->Text = toSysString(op1.toString());

lSecondOutput->Text = toSysString(op2.toString());

lFirstOutput->Text = toSysString(op1.toString());

}

catch (EDivisionByZero eDivisionByZero)

{

MessageBox::Show(eDivisionByZero.toString(), "Error");

}

}

}

private: System::Void bArg\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put)

{

op1 = op1.arg();

lResultOutput->Text = toSysString(op1.toString());

lSecondOutput->Text = toSysString(op2.toString());

lFirstOutput->Text = toSysString(op1.toString());

}

}

private: System::Void bConj\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put)

{

op1 = op1.conjugate();

lResultOutput->Text = toSysString(op1.toString());

lSecondOutput->Text = toSysString(op2.toString());

lFirstOutput->Text = toSysString(op1.toString());

}

}

private: System::Void bAbs\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put)

{

op1 = op1.abs();

lResultOutput->Text = toSysString(op1.toString());

lSecondOutput->Text = toSysString(op2.toString());

lFirstOutput->Text = toSysString(op1.toString());

}

}

private: System::Void bSwap\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put && op2Put)

{

Complex tmp;

tmp = op1;

op1 = op2;

op2 = tmp;

lSecondOutput->Text = toSysString(op2.toString());

lFirstOutput->Text = toSysString(op1.toString());

}

}

private: System::Void bEquals\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put && op2Put)

{

if (op1.equals(op2))

lResultOutput->Text = "True";

else

lResultOutput->Text = "False";

}

}

private: System::Void bExp\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put)

{

op1 = op1.compExp();

lResultOutput->Text = toSysString(op1.toString());

lSecondOutput->Text = toSysString(op2.toString());

lFirstOutput->Text = toSysString(op1.toString());

}

}

private: System::Void bCos\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put)

{

res = op1.cos();

lResultOutput->Text = res.ToString();

lSecondOutput->Text = toSysString(op2.toString());

lFirstOutput->Text = toSysString(op1.toString());

}

}

private: System::Void bSin\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put)

{

res = op1.sin();

lResultOutput->Text = res.ToString();

lSecondOutput->Text = toSysString(op2.toString());

lFirstOutput->Text = toSysString(op1.toString());

}

}

private: System::Void bSinh\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put)

{

res = op1.sinh();

lResultOutput->Text = res.ToString();

}

}

private: System::Void bCosh\_Click(System::Object^ sender, System::EventArgs^ e) {

if (op1Put)

{

res = op1.cosh();

lResultOutput->Text = res.ToString();

}

}

//Menu strip realization

private: System::Void helpToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageBox::Show("To use this program enter complex numbers and do operations with them", "Help info");

}

private: System::Void aboutToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageBox::Show("This program was made as a course work \nBy student of PZKS IFTKN Chernivtsi National University \nAt 2013 \nRecieved the highest mark",

"About info");

}

private: System::Void exitToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

exit(0);

}

private: System::Void whiteToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

this->BackColor = System::Drawing::Color::White;

}

private: System::Void blackToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

this->BackColor = System::Drawing::Color::LightBlue;

}

private: System::Void greyToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

this->BackColor = System::Drawing::Color::LightGray;

}

private: System::Void coralToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

this->BackColor = System::Drawing::Color::Coral;

}

private: System::Void redToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

this->BackColor = System::Drawing::Color::LightSalmon;

}

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void greenToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

this->BackColor = System::Drawing::Color::LightGreen;

System::Drawing::Color col;

col = System::Drawing::Color::FromArgb(152, 251, 152);

this->BackColor = col;

}

private: System::Void toolStripMenuItem2\_Click(System::Object^ sender, System::EventArgs^ e) {

System::Drawing::Color col;

col = System::Drawing::Color::FromArgb(255, 250, 240);

this->BackColor = col;

}

private: System::Void toolStripMenuItem3\_Click(System::Object^ sender, System::EventArgs^ e) {

System::Drawing::Color col;

col = System::Drawing::Color::FromArgb(240, 255, 240);

this->BackColor = col;

}

private: System::Void toolStripMenuItem4\_Click(System::Object^ sender, System::EventArgs^ e) {

System::Drawing::Color col;

col = System::Drawing::Color::FromArgb(30, 144, 255);

this->BackColor = col;

}

private: System::Void toolStripMenuItem5\_Click(System::Object^ sender, System::EventArgs^ e) {

System::Drawing::Color col;

col = System::Drawing::Color::FromArgb(32, 178, 170);

this->BackColor = col;

}

private: System::Void menuStrip1\_ItemClicked(System::Object^ sender, System::Windows::Forms::ToolStripItemClickedEventArgs^ e) {

}

//System language: Ukrainian

private: System::Void ukrToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

localize.setCulture("Ua");

refresh();

}

private: System::Void englishToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

localize.setCulture("Eng");

refresh();

/\*ofstream of("Eng.clt");

of << "Change first" << '\n';

of << "Change second" << '\n';

of << "equals" << '\n';

of << "arg" << '\n';

of << "exp" << '\n';

of << "conj" << '\n';

of << "abs" << '\n';

of << "real" << '\n';

of << "imagine" << '\n';

of << "result" << '\n';

of << "first" << '\n';

of << "second" << '\n';

of << "none" << '\n';

of << "Operations with both operands" << '\n';

of << "Operations with one argument" << '\n';

of << "File" << '\n';

of << "Language" << '\n';

of << "Exit" << '\n';

of << "Themes" << '\n';

of << "White" << '\n';

of << "Blue" << '\n';

of << "Green" << '\n';

of << "Coral" << '\n';

of << "Salmon" << '\n';

of << "Help..." << '\n';

of << "About..." << '\n';

of.close();\*/

}

};

}

//MyForm.cpp

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void main(){

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

CourseWork::MyForm form;

Application::Run(%form);

}