In main.cpp:

#include <iostream>

#include "Complex.h"

int main() {

Complex num1 (4, 6);

Complex num2 (2, 3);

Complex temp = -num1;

temp.printNumber();

temp = num1 + num2;

temp.printNumber();

temp = num1 - num2;

temp.printNumber();

temp = num1 \* num2;

temp.printNumber();

temp = num1 / num2;

temp.printNumber();

temp = num1 + 2;

temp.printNumber();

temp = num1 - 2;

temp.printNumber();

temp = num1 \* 2;

temp.printNumber();

temp = num1 / 2;

temp.printNumber();

return 0;

}

В Complex.h:

#ifndef \_\_COMPLEX\_\_HEADER

#define \_\_COMPLEX\_\_HEADER

class Complex {

public:

//Constructor

Complex(int real, int img);

//Copy constructor

Complex(const Complex& rhs);

//Оператор за присвояване

Complex& operator=(const Complex& rhs);

//Destructor

~Complex();

//The defaulty generated copy constr and operator =

//Work very well, but i will overrite them just to be sure

void printNumber();

// Operator for arithmetic negation

Complex operator-() const;

// Operators for adding, removing, ...

// to the current value, thats why it returns

// the object with амперсант

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

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

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

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

//Same operators but with real number

//Instead of a complex one

Complex& operator+=(const int& num);

Complex& operator-=(const int& num);

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

Complex& operator/=(const int& num);

private:

void copy(const Complex& rhs);

private:

int real;

int imag;

};

//single operators +, -, \*, /, should be outside the class

//because this operators, do not belong to either of the numbers(objects)

// e.g.: a + b creates new objects which contain the sum of the two

//Thats why it returns "Complex", and not "Complex&", and its not in the class

//Its also const, because "a + b = c" should not be possible

const Complex operator+(const Complex& num1, const Complex& num2);

const Complex operator-(const Complex& num1, const Complex& num2);

const Complex operator\*(const Complex& num1, const Complex& num2);

const Complex operator/(const Complex& num1, const Complex& num2);

//Same operators but with a real number as a second argument

//Instead of a complex

const Complex operator+(const Complex& num1, const int& num2);

const Complex operator-(const Complex& num1, const int& num2);

const Complex operator\*(const Complex& num1, const int& num2);

const Complex operator/(const Complex& num1, const int& num2);

#endif

В Complex.cpp:

#include "Complex.h"

#include <iostream>

using std::cout;

using std::endl;

Complex::Complex(int \_real, int \_imag){

real = \_real;

imag = \_imag;

}

Complex::Complex(const Complex& rhs){

copy(rhs);

}

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

//This checks if the addresses of the current object

//and the passed object are the same

if(this != & rhs){

copy(rhs);

}

return \*this;

}

//No dynamic memory so default destructor is used

Complex::~Complex(){

}

void Complex::copy(const Complex& rhs){

real = rhs.real;

imag = rhs.imag;

}

void Complex::printNumber(){

cout << real;

if (imag >= 0){

cout << " +i " << imag;

} else {

cout << " -i" << imag;

}

cout << endl;

}

Complex Complex::operator-() const{

return Complex(-real, -imag);

//Връщаме нов обект, на който е с обратни стойности на числата

}

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

real += num.real;

imag += num.imag;

return \*this;

// Понеже имаме референция връщаме съшия обект

// За когото в викнат оператора

}

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

real -= num.real;

imag -= num.imag;

return \*this;

}

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

real \*= num.real;

imag \*= num.imag;

return \*this;

}

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

real /= num.real;

imag /= num.imag;

return \*this;

}

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

real += num;

return \*this;

}

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

real -= num;

return \*this;

}

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

real \*= num;

return \*this;

}

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

real /= num;

return \*this;

}

const Complex operator+(const Complex& num1, const Complex& num2){

return Complex(num1) += num2;

//Връщаме нов обект, който е със същите стойности като num1

// Обаче събран с числата на втория подаден обект.

// Преизползваме оператора += , който сме написали по горе

}

const Complex operator-(const Complex& num1, const Complex& num2){

return Complex(num1) -= num2;

}

const Complex operator\*(const Complex& num1, const Complex& num2){

return Complex(num1) \*= num2;

}

const Complex operator/(const Complex& num1, const Complex& num2){

return Complex(num1) /= num2;

}

const Complex operator+(const Complex& num1, const int& num2){

return Complex(num1) += num2;

}

const Complex operator-(const Complex& num1, const int& num2){

return Complex(num1) -= num2;

}

const Complex operator\*(const Complex& num1, const int& num2){

return Complex(num1) \*= num2;

}

const Complex operator/(const Complex& num1, const int& num2){

return Complex(num1) /= num2;

}