16.10.2020

Sprawozdanie
Programowanie Komputerów 3
Laboratorium 2

Paulina Czapla
Informatyka AEI,sem. 3, gr.4

Zespolona.h

```
#include <iostream>



class Zespolona
{
public:
	float IM;
	float RE;
	static int licznik;


	Zespolona(float, float);
	Zespolona(float);
	Zespolona();
	Zespolona(Zespolona&);
	~Zespolona();
	void set_IM(float);
	void set_RE(float);
	float get_IM();
	float get_RE();
	void wyswietl();
	void pokaz_ile_obiektow();


	Zespolona& operator+=(const Zespolona&);
	Zespolona& operator+ (const Zespolona&) const;
	Zespolona& operator- (const Zespolona&) const;
	const float& operator[] (bool) const;
	bool operator!=(const Zespolona&);
	std::istream& operator>> (std::istream&);
	std::ostream& operator<< (std::ostream& output);
	Zespolona& operator=(const Zespolona&);
	bool operator==(const Zespolona&);
	Zespolona& operator ++();

	Zespolona& operator+= (float);
	//Zespolona& operator+= (float);
	//float operator+= (float);
	Zespolona& operator() (float, float);

};
```

definicje operatorów z pliku Zespolona.cpp

```
Zespolona& Zespolona::operator+= (const Zespolona& z2)
{
	std::cout << " Zespolona& Zespolona::operator+= (const Zespolona& z2) " << std::endl;
	this->IM = this->IM + z2.IM;
	this->RE = this->RE + z2.RE;

	return *this;
}

Zespolona& Zespolona::operator+ (const Zespolona& z2) const
```

```
{
	std::cout << " Zespolona& Zespolona::operator+ (const Zespolona& z2) const " <<
std::endl;
	Zespolona* z3 = new Zespolona(this->IM + z2.IM, this->RE + z2.RE);
	return *z3;
}

Zespolona& Zespolona::operator- (const Zespolona& z2) const
{
	std::cout << " Zespolona& Zespolona::operator- (const Zespolona& z2) const " <<
std::endl;
	Zespolona* z3 = new Zespolona(this->IM - z2.IM, this->RE - z2.RE);

	return *z3;
}

const float & Zespolona::operator[](bool n) const
{
	std::cout << " const float & Zespolona::operator[](bool n) const " << std::endl;
	if (n)
		return this->RE;
	else
		return this->IM;

}

bool Zespolona::operator!=( const Zespolona& z2)
{
	std::cout << " bool Zespolona::operator!=( const Zespolona& z2) " << std::endl;
	if (this->IM == z2.IM && this->RE == z2.RE)
		return false;
	else true;
}


std::istream& Zespolona::operator>> (std::istream& input)
{
	std::cout << " std::istream& Zespolona::operator>> (std::istream& input) " <<
std::endl;
	float a, b;
	input >> a >> b;
	this->RE = a;
	this->IM = b;
	return input;
}

std::ostream& Zespolona::operator<< (std::ostream& output)
{
	std::cout << " std::ostream& Zespolona::operator<< (std::ostream& output) " <<
std::endl;
	output << "Liczba rzeczywista: " << this->RE << " liczba zespolona: " << this->IM <<
std::endl;
	return output;
}


Zespolona& Zespolona::operator=(const Zespolona& z)
{
	std::cout << " Zespolona& Zespolona::operator=(const Zespolona& z) " << std::endl;
	this->RE = z.RE;
	this->IM = z.IM;
	return *this;

}
```

```
bool Zespolona::operator==(const Zespolona& z2)
{
	std::cout << " bool Zespolona::operator==(const Zespolona& z2) " << std::endl;
	if (this->IM == z2.IM && this->RE == z2.RE)
		return true;
	else false;
}


Zespolona& Zespolona::operator++()
{
std::cout << " Zespolona& Zespolona::operator++() " << std::endl;
	this->RE = this->RE + 1;
	return *this;
}

Zespolona& Zespolona::operator+= (float num)
{
	std::cout << " Zespolona& Zespolona::operator+= (float num) " << std::endl;
	this->IM = this->IM + num;
	this->RE = this->RE + num;
	return *this;
}

//Zespolona& Zespolona::operator+= (float num)
//{
//std::cout << " Zespolona& Zespolona::operator+= (float num)" << std::endl;
//	this->IM = this->IM + num;
//
//	return *this;
//}
//
//float Zespolona::operator+= (float num)
//{
//std::cout << " float Zespolona::operator+= (float num) " << std::endl;
//	num += this->get_RE;
//
//	return num;
//}

Zespolona& Zespolona::operator() (float re, float im)
{
	this->RE = re;
	this->IM = im;
	return *this;
}
```

main.cpp

```
#include "zespolona.h"
#include <iostream>



Zespolona& operator+(const Zespolona& z1, const Zespolona& z2)
{
    Zespolona* z3 = new Zespolona (z1.IM + z2.IM, z1.RE+z2.RE);
    return *z3;
}

Zespolona& operator-(const Zespolona& z1,const Zespolona& z2)
{
    Zespolona* z3 = new Zespolona(z1.IM - z2.IM, z1.RE - z2.RE);
```

```
    return *z3;
}

int main()
{
    Zespolona z1 (2, 3);
    Zespolona  z2 (3, 3);
    std::cout << z1;
    z1 += z2;
    z1.wyswietl();
    Zespolona z3 = z1 + z2;
    z3.wyswietl();
    if (z1 != z2)
        std::cout << z1;
    Zespolona z4;
    std::cin >> z4;
    z4.wyswietl();
    std::cout << z4[1]<<"   "<<z4[0]<<std::endl;

}
```

globalne implementacje niektórych operatorów:

```
bool operator !=(const Zespolona& z1,const Zespolona& z2)
{
    if (z1.IM == z2.IM && z1.RE == z2.RE)
        return false;
    else true;
}

bool operator ==(const Zespolona& z1, const Zespolona& z2)
{
    if (z1.IM == z2.IM && z1.RE == z2.RE)
        return true;
    else false;
}

std::istream& operator>> (std::istream& input, Zespolona& z)
{
    float a, b;
    input >> a >> b;
    z.set_RE(a);
    z.set_IM(b);

    return input;
}

std::ostream& operator<< (std::ostream& output, Zespolona const& z)
{
    output << "Liczba rzeczywista: " << z.RE << " liczba zespolona: " << z.IM << std::endl;
    return output;
}

Zespolona& operator++(Zespolona& z)
{
    z.RE = z.RE + 1;
    return z;
}
```