Laboratorium 9 Pomoc. Przeciążanie operatorów

Cel laboratorium:
Zapoznanie z ideą i sposobem realizacji przeciążania operatorów

- Przeciążanie/przeładowanie operatorów: określanie ***nowych działań*** (zmiana znaczenia) wybranych operatorów na potrzeby ***danej klasy =>*** usprawnienie pracy na obiektach
- Sposób realizacji - zdefiniowanie funkcji operatorowej:

```
typ_zwracany operator <symbol> (<opcjonalnie parametry>)
{<instrukcje>}
```

- Operatory, które można przeciążać:
  + - * / % ^ & ! = < > == ++ -- != >= , -> [] new delete itp.
- Operatory, które nie można przeciążać:
  . (dostęp do składowych obiektu), .* (wskazanie na element obiektu), :: (operator zakresu), ?: (operator warunkowy), #, ##
- Funkcja operator <symbol> musi posiadać co najmniej 1 argument danej klasy -> nie można zmieniać działania operatorów dla wbudowanych typów: int, float, ...
- Argumenty funkcji operator <symbol>: obiekt danej klasy lub referencja do obiektu (nie może być wskaźnik do obiektu)
- Funkcja **operator <symbol>** może być :
  - ***zwykłą funkcją*** (np. globalną) – pracuje na publicznych składnikach klasy
  - ***zaprzyjaźnioną funkcją klasy*** – pracuje na prywatnych składnikach klasy
  - ***niestatyczną metodą klasy*** (ma o 1 parametr mniej – niejawny wskaźnik this; występuje obowiązkowo dla operatorów: =, {}, (), ->)

Przykład - klasa Tzespolona do wykonywania operacji na liczbach zespolonych a+bi

Tzespolona.h

```
#ifndef TZESPOLONA_H
#define TZESPOLONA_H
#include  <iostream>
using namespace std;
class Tzespolona
{public:
   Tzespolona(double rz, double ur): rzeczywista(rz),
urojona(ur){ };
   ~Tzespolona();
   Tzespolona operator+(Tzespolona b); //p.o. - metoda klasy
   friend ostream & operator<<(ostream & os, const
        Tzespolona & z);//p.o. - funkcja zaprzyjazniona
   double rzeczywista;//publiczne dla operator-
   double urojona;
};
#endif
```

Tzespolona.cpp

```
#include "Tzespolona.h"

Tzespolona::~Tzespolona()
{
}
Tzespolona Tzespolona::operator+(Tzespolona b)
{Tzespolona suma(0,0);
  suma.rzeczywista=this->rzeczywista+b.rzeczywista;
  suma.urojona=this->urojona +b.urojona;
  return suma;
}
```

main.cpp

```
#include <iostream>
#include "Tzespolona.h"
using namespace std;

Tzespolona operator-(Tzespolona a, Tzespolona b);
//=========================
int main(int argc, char** argv) {
   Tzespolona z1(1,2.4), z2(6.9,2.3), z3(0,0);
   cout<<"z1="<<z1<<endl<<" z2="<<z2<<endl;
   z3=z1+z2;
   cout<<"suma z1+z2="<<z3<<endl;
   z3=z1-z2;
   cout<<"roznica z1-z2="<<z3<<endl;
   return 0;
}
//=========================globalna f.=============
Tzespolona operator-(Tzespolona a, Tzespolona b
{Tzespolona roznica(0,0);
  roznica.rzeczywista=a.rzeczywista-b.rzeczywista;
  roznica.urojona=a.urojona -b.urojona;
  return roznica;
}
//=========================globalna f.-przyjaciel===
ostream & operator<<(ostream & os, const Tzespolona & z)
{os<<z.rzeczywista << "+" << z.urojona << "i";
return os;

}
```