Temat: Przeciążanie operatorów w języku C++

Operatory:

- Przeciążanie operatorów pozwala redefiniować operator dla elementów klasy (lub struktury).
- Operatory, które mogą zostać przeciążone:
  - Dwuargumentowe: +, *, /, %, ^, &, |, <<, >>
  - Jednoargumentowe: +, -, *, &
  - Relacji: ==, !=, <, >, <=, =>
  - Przypisania: +=, -=, *=, /=, %=, |=, <<=, >>=
  - Inkrementacji dekrementacji: ++, --
  - Logiczne: ||, &&, !
  - Wejścia wyjścia: <<, >>
  - Inne: [], (), ...

- Różne typy operatorów przeciąża się na różne sposoby.

- Dokumentacja:
  https://en.cppreference.com/w/cpp/language/operators

Przeciążanie operatorów:

```cpp
#include <iostream>
using namespace std;

class licz
{
public:
    int x,y;
    licz(): x(0), y(0) {}
    licz (int a,  int b) : x(a), y(b) {}

    friend ostream& operator << (ostream&, const licz&);
};

licz operator + (licz a, licz b)
{
    licz c;
    c.x = a.x+b.x;
    c.y = a.y+b.y;
    return c;
    //return licz(a.x+b.x, a.y+b.y);
}
bool operator > (licz a, licz b){return ((a.x + a.y) > (b.x + b.y) );}
licz operator - (licz &a){return licz(-a.x, -a.y);}
licz operator -- (licz &a){return licz(--a.x, --a.y);}
licz operator ++ (licz &a, int){return licz(a.x++, a.y++);}
licz operator -= (licz &a, licz &b){return licz(a.x-=b.x, a.y-=b.y);}

ostream& operator << (ostream& s, const licz& l){return s<<"["<<l.x<<", "<<l.y<<"]";}

int main()
{
    licz a(11,12), b(3,3),c(1,1);
    cout<<"a+b="<<a+b<<endl;
    if (a>b) cout<<"a+b+c="<<a+b+c<<endl;

    return 0;
}
```

```
a+b=[14, 15]
a+b+c=[15, 16]
```

Zadania do samodzielnej realizacji:

(2p) Wykonaj ponownie Zadanie 2 – Laboratorium 10 z tą różnicą, że zamiast struktury należy użyć klasy, zamiast funkcji *suma*, *roznica*, *iloczyn* należy przeciążyć operatory „**+**”, „**-**”, „*****” oraz metodę wypisz() przez operator „<<” realizujące analogiczne operacje. Przetestuj działanie przeciążonych operatorów. Wyniki w sposób czytelny wyświetl na ekranie.

(3p) Zaimplementuj klasę Zesp symulującą działania na liczbach zespolonych. Utwórz konstruktor, destruktor. Przeciąż 8 operatorów z "*", "/", "-", "+", "*=", "=", "/=", "-=", "+=", "--", "++", "<", ">" (porównywanie modułów), "==", "<<" (wypisywanie do strumienia). Napisać kod testujący działanie każdego przeciążonego operatora.