ЛАБОРАТОРНА РОБОТА 5
ПЕРЕВАНТАЖЕННЯ ОПЕРАЦІЙ

Мета роботи: Отримати практичні навички створення абстрактних типів даних і перевантаження операцій в мові С++.

Основні теоретичні відомості

У С++ оператори можна перевантажувати для типів даних, які визначаються програмістом. Принциповий виграш від перевантаження операторів полягає в тому, що воно дозволяє органічно інтегрувати нові типи даних в середовище програмування.

Перевантаження операторів тісно пов'язане з перевантаженням функцій. Щоб перевантажити оператор, необхідно визначити значення нової операції для класу, до якого вона буде застосовуватися. Для цього створюється функція `operator` (операційна функція), яка визначає дію цього оператора. Загальний формат функції operator такий:

тип ім'я_класу :: `operator` # (список аргументів)

{

операція_над_класом

}

Оператори перевантажуються за допомогою функції `operator.`

Тут оператор, що перевантажується, позначається символом "#", а елемент тип являє собою тип значення, що повертається заданої операцією.

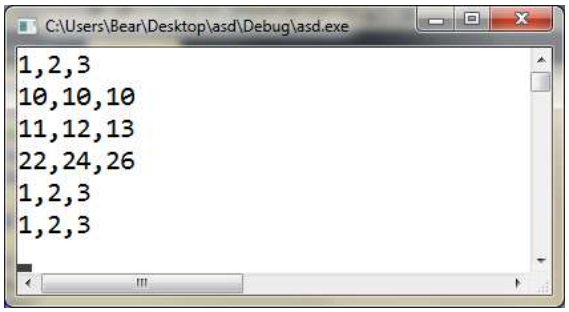
Функція-оператор може бути членом класу чи не бути їм. Операторні функції, які не є членами класу, часто визначаються як його «друзі». Операторні функції-члени і функції, які не є членами класу, розрізняються за формою перевантаження. Кожен з варіантів ми розглянемо окремо.

Перевантаження операторів з використанням функцій-членів класу

У наступній програмі створюється клас three_d, який підтримує координати об'єкта в тривимірному просторі. Для класу three_d перевантажуються оператори + і =. Отже, розглянемо уважно код цієї програми.

```
// Перевантаження операторів за допомогою функцій-членів.
#include <iostream>
#include <conio.h>
using namespace std;

class three_d {
int x, y, z; // 3-мірні координати
public:
three_d () {x = y = z = 0; }
three_d (int i, int j, int k) {x = i; y = j; z = k; }
three_d operator + (three_d op2); // Операнд op1
                                  //передається неявно.
three_d operator = (three_d op2); // Операнд op1
                                  //передається неявно.
void show ();
};
// Перевантаження оператора "+".
three_d three_d :: operator + (three_d op2)
{
three_d temp;
temp.x = x + op2.x; // Операції додавання цілочисельних
temp.y = y + op2.y; // значень зберігають оригінальний
temp.z = z + op2.z; // сенс.
return temp;}
// Перевантаження оператора присвоювання.
three_d three_d :: operator = (three_d op2)
{
x = op2.x; // Операції присвоювання цілочисельних
y = op2.y; // значень зберігають оригінальний
z = op2.z; // сенс.
return * this;
}
// Відображення координат X, Y, Z.
void three_d :: show ()
{
cout << x << ",";
cout << y << ",";
cout << z << endl;
}

int main ()
{
three_d a (1, 2, 3), b (10, 10, 10), c;
a.show ();
b.show ();
c = a + b; // додавання об'єктів a й b
c.show ();
c = a + b + c; // додавання об'єктів a, b і з
c.show ();
c = b = a; // демонстрація множинного присвоювання
c.show ();
b.show ();
}
```

При виконанні ця програма генерує такі результати



При перевантаженні бінарного оператора з використанням функції-члена їй передається явно тільки один аргумент. Другий же неявно передається через this. Таким чином, в рядку

temp.x = x + op2.x;

Під членом x мається на увазі член this-> x, тобто член x зв'язується з об'єктом, який викликає дану операторну функцію. У всіх випадках неявно передається об'єкт, що вказується зліва від символу операції, який став причиною виклику операторної функції. Об'єкт, що розташовується з правого боку від символу операції, передається цій функції як аргумент. У загальному випадку при використанні функції-члена для перевантаження унарного оператора параметри не використовуються взагалі, а для перевантаження бінарного – тільки один параметр. *Тернарний оператор «?» перевантажувати не можна.*

Зверніть увагу на те, що функція operator + () повертає об'єкт типу three_d. Незважаючи на те що вона могла б повертати значення будь-якого допустимого в C++ типу, той факт, що вона повертає об'єкт типу three_d, дозволяє використовувати оператор "+" в таких складових виразах, як a + b + c. Частина цього виразу, a + b, генерує результат типу three_d, який потім підсумовується з об'єктом c. І якби ця частина виразу генерувала значення іншого типу (а не типу three_d), такий складний вираз просто не працював би.

На відміну від оператора "+", оператор присвоювання призводить до модифікації одного зі своїх аргументів. (Перш за все, це становить саму суть присвоювання.) Оскільки функція operator = () викликається об'єктом, який розташований зліва від символу присвоювання (=), саме цей об'єкт і

модифікується в результаті операції присвоювання. Після виконання цієї операції значення, що повертається перевантаженим оператором, містить об'єкт, який розташований зліва від символу присвоювання. (Такий стан речей цілком узгоджується з традиційною дією оператора "=".) Наприклад, щоб можна було виконувати інструкції, подібні наступної:

a = b = c = d;

Необхідно, щоб функція operator = () повертала об'єкт, що адресується вказівником *this*, і щоб цей об'єкт був розташований зліва від оператора "=". Це дозволить виконати будь-який ланцюжок присвоювання. Операція присвоювання – це одне з найважливіших застосувань вказівника *this*.

Можна також перевантажувати унарні оператори таки, як "++", "–", або унарні "–" і "+". Як згадувалося вище, при перевантаженні унарних оператора за допомогою функції-члена операторної функції жоден об'єкт не передається явно. Операція ж виконується над об'єктом, який генерує виклик цієї функції через неявно переданий покажчик this.

Перевантаження операторів з використанням функцій, яки не є членами классу

Перевантаження оператора для класу можна реалізувати і з використанням функції, що не є членом цього класу. Такі функції часто визначаються «друзями» класу. Бінарні операторні функції, які не є членами класу, мають два параметри, а унарні - один. Як згадувалося вище, функції-не члени (в тому числі і функції-«друзі») не мають покажчика this. Отже, якщо для перевантаження бінарного оператора використовується функція-«друг», явно передаються обидва операнда. З використанням функцій-не членів класу не можна перевантажувати такі оператори:

`=, (), [] , ->.`

Наприклад, в наступній програмі для перевантаження оператора + замість функції-члена використовується функція-«друг».

```
// Перевантаження оператора "+" за допомогою функції-"друга".
#include <iostream>
#include <conio.h>
using namespace std;

class three_d {
int x, y, z; // 3-мірні координати
public:
three_d () {x = y = z = 0; }
three_d (int i, int j, int k) {x = i; y = j; z = k; }
friend three_d operator + (three_d op1, three_d op2);
three_d operator = (three_d op2); // op1 передається неявно.
void show ();
};

// Тепер це функція-"друг".
three_d operator + (three_d op1, three_d op2)
{
three_d temp;
temp.x = op1.x + op2.x;
temp.y = op1.y + op2.y;
temp.z = op1.z + op2.z;
return temp;
}
// Перевантаження присвоювання.
three_d three_d :: operator = (three_d op2)
{
x = op2.x;
y = op2.y;
z = op2.z;
return * this;
}
// Відображення координат X, Y, Z.
void three_d :: show ()
{
cout << x << ",";
cout << y << ",";
cout << z << "";
}

int main ()
{
three_d a (1, 2, 3), b (10, 10, 10), c;
a.show ();
b.show ();
c = a + b; // додавання об'єктів a й b
c.show ();
c = a + b + c; // додавання об'єктів a, b і з
c.show ();
c = b = a; // демонстрація множинного присвоювання
c.show ();
b.show ();
}
```

Операторної функції operator + () тепер передаються два операнда. Лівий операнд передається параметру op1, а правий - параметру op2.

Для реалізації перевантаження операторів слід використовувати функції-члени. З використанням функцій-не членів класу не можна перевантажувати такі оператори: =, (), [] , ->.

Функції-«друзі» використовуються в С++ в основному для обробки спеціальних деталей і ситуацій.

Перевантаження операторів відношення і логічних операторів

Оператори відношення (наприклад, == або <) і логічні оператори (наприклад, && або | |) також можна перевантажувати, причому робити це зовсім неважко. Як правило, перевантажена операторна функція відношення повертає об'єкт класу, для якого вона перевантажується. А перевантажений оператор відношення або логічний оператор повертає одне з двох можливих значень: `true` або `false`. Це відповідає звичайному застосуванню цих операторів і дозволяє використовувати їх в умовних виразах.

Розглянемо приклад перевантаження оператора "==" для вже знайомого нам класу `three_d`.

```
// Перевантаження оператора "=="
bool three_d :: operator == (three_d op2)
{
if ((x == op2.x) && (y == op2.y) && (z == op2.z))
return true;
else
return false;
}
```

Якщо вважати, що операторна функція operator == () вже реалізована, наступний фрагмент коду абсолютно коректний.

```
three_d a, b;
// ...
if(a == b)
 cout << "a дорівнює b";
else
 cout << "a не дорівнює b";
```

Перевантаження оператора індексації масивів ([])

Крім традиційних операторів C++ дозволяє перевантажувати і більш «екзотичні», наприклад, оператор індексації масивів ([ ]). У C++ (з точки зору механізму перевантаження) оператор [ ] вважається бінарним. Його можна перевантажувати тільки для класу і тільки з використанням функції-члена. Ось як виглядає загальний формат операторної функції-члена operator [ ] ().

```
тип ім'я_класу :: operator [] (int індекс)
{
// ...
}
```

Припустимо, у нас визначено об'єкт ob, тоді вираз

`ob[3]` перетвориться в наступний виклик операторної функції operator []():

```
ob.operator [] (3)
```

Іншими словами, значення виразу, заданого в операторі індексації, передається операторної функції operator [] () в якості явно заданого аргументу. При цьому покажчик this буде вказувати на об'єкт ob, тобто об'єкт, який генерує виклик цієї функції.

У наступній програмі в класі atype оголошується масив для зберігання трьох int-значень. Його конструктор ініціалізує всі члени цього масиву. Перевантажена операційна функція operator [] () повертає значення елемента, заданого його параметром.

```
// Перевантаження оператора індексації масивів
#include <iostream>
#include <conio.h>
using namespace std;
const int SIZE = 3;
class atype {
int a [SIZE];
public:
atype () {
register int i;
for (i = 0; i <SIZE; i ++) a [i] = i;
}
int operator [] (int i) {return a [i];}
};
```

```
int main ()
{
atype ob;
cout << ob [2]; // відображає число 2
}
```

Тут функція operator [] () повертає значення і-го елемента масиву A. Таким чином, вираз ob [2] повертає число 2, яке відображається інструкцією cout. Ініціалізація масиву a за допомогою конструктора (в цьому і наступному програмах) виконується лише в ілюстративних цілях.

Одна з переваг перевантаження оператора "[]" полягає в тому, що з його допомогою ми можемо забезпечити засіб реалізації безпечної індексації масивів.

Механізм перевантаження операторів можна використовувати для додавання нових типів даних в середовище програмування. Це один з найбільш потужних засобів C ++.

Приклад розв'язання завдань

Створити клас «Комплексне число».

1. Операція «+» перевантажена: для двох комплексних чисел створює нове комплексне число, дійсна і уявнв частина якого дорівнює сумі дійсних і уявних частин двох комплексних доданків.

2. Операція «++» перевантажена як префіксна так і постфіксна: збільшує на одиницю дійсну і уявну частину комплексного числа.

3. Операція << перевантажена і виводить на екран значення дійсної і уявної частини комплексного числа.

Визначення класу `Complex:`

```
class Complex // початок оголошення класу
{
 double real;  // дійсна частина
 double imag;  // уявна частина
public:
 Complex()
 {    real = 0;
      imag = 0;
 }
```

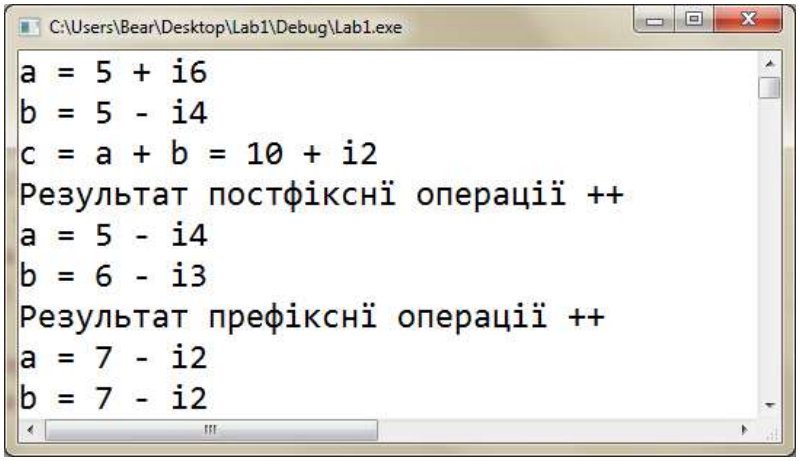
```
 Complex(double real, double imag)
 {
      this->real = real;
      this->imag = imag;
 }
 Complex operator + (Complex &obj)
 {
      Complex t(real + obj.real, imag + obj.imag);
      return t;
 }
 // Перевантаження префіксной версії оператора "++".
 Complex operator ++()
 {
      real++; // інкремент дійсної частини
      imag++; // інкремент уявної частини
      return *this;
 }

 // Перевантаження постфіксной версії оператора "++".
 Complex operator ++(int notused)
 {
      Complex t = *this; // збереження вихідного значення
      real++; // інкремент дійсної частини
      imag++; // інкремент уявної частини
      return t; // повернення початкового значення
 }
 friend ostream& operator  << (ostream &t, Complex &obj);
}; // кінець оголошення класу

// дружня функція-оператор <<
ostream& operator  << (ostream &t, Complex &obj)
{
 if (obj.imag < 0)
      t << obj.real << " - i" << -(obj.imag);
 else
      t << obj.real << " + i" << obj.imag;
 return t;
}
```

Головна функція main:

```
int main() {
 Complex a(5, 6), b(5, -4);
 Complex c = a + b;
 //
 cout << "a = " << a << endl << "b = " << b << endl;
 cout << "c = a + b = " << c << endl;
 a = b++;
 cout<<"Результат постфіксні операції ++" << endl;
```

```
 cout << "a = " << a << endl << "b = " << b << endl;
 a = ++b;
 cout << "Результат префіксні операції ++" << endl;
 cout << "a = " << a << endl << "b = " << b << endl;
}
```

Результат роботи програми:



Завдання до лабораторної роботи:

1. Визначити і реалізувати клас – абстрактний тип даних.

2. Визначити і реалізувати операції над даними цього класу відповідно варіанту.

3. Для всіх варіантів при перевантаженні операторів «інкремент» (++) або «декремент» (--) реалізувати як префіксну, так і постфіксну форму.

4. Для всіх варіантів перевантажити оператор << для виводу на консоль інформації про об’єкт.

Варіанти завдань

Варіант 1.

Створити клас «квадрат», такий, що:

a) його екземпляр містить розмір сторони квадрата.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням сторони.

c) його методи дозволяють отримувати і присвоювати значення сторони.

d) операція «<<» перевантажена: виводить на екран значення сторони і площі квадрата.

e) операція «+» перевантажена: для двох квадратів створює новий квадрат сумарної площі. Значення сторони нового квадрата має бути перераховане.

g) операція «++» збільшує стронону квадрата на одиницю.

Варіант 2.

Створити клас «трикутник», такий, що:

a) його екземпляр містить розміри трьох сторін трикутника.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням сторін.

c) його методи дозволяють отримувати і присвоювати значення сторін і обчислювати площу (наприклад, за формулою Герона), $s = \sqrt{p(p-a)(p-b)(p-c)}$, $p = (a+b+c)/2$.

d) операція «<<» перевантажена: виводить на екран значення трьох сторін і площі трикутника.

e) операція «+» перевантажена: для двох трикутників створює новий трикутник, сторони якого дорівнюють сумам відповідних сторін цих двох трикутників.

g) операція «++» збільшує всі стронони трикутника на одиницю.

Варіант 3

Створити базовий клас «циліндр», такий, що:

a) його екземпляр містить розмір радіуса і висоти.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням радіуса і висоти.

c) його методи дозволяють отримувати і присвоювати значення радіуса і висоти.

d) операція «<<» перевантажена: виводить на екран значення радіуса, висоти та обчислювати значення об’єму циліндра.

e) операція «+» перевантажена: для двох циліндрів створює новий циліндр об’єм якого дорівнює сумі об’ємів цих двох циліндрів, а радіус дорівнює максимальному серед цих двох циліндрів.

g) операція «++» збільшує радіус і висоту циліндра на одиницю.

Варіант 4

Створити клас «куля», такий, що:

a) його екземпляр містить розмір радіусу.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням радіусу.

c) його методи дозволяють отримувати і присвоювати значення радіусу і обчислювати значення об'єму.

d) операція «<<» перевантажена: виводить на екран значення радіуса і об'єму кулі.

e) операція «+» перевантажена: для двох куль створює нову кулю сумарного об’єму. Радіус нової кулі має бути перевизначений.

g) операція «++» збільшує радіус кулі на одиницю.

Варіант 5.

Створити клас «конус», такий, що:

a) його екземпляр містить розмір радіуса і висоти.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням радіуса і висоти.

c) його методи дозволяють отримувати і присвоювати значення радіуса і висоти та обчислювати значення об'єму.

d) операція «<<» перевантажена: виводить на екран значення радіуса, висоти і об’єму конуса.

e) операція «+» перевантажена: для двох конусів створює новий конус з параметрами r = r1 + r2, h = h1 + h2.

g) операція «++» збільшує радіус і висоту конуса на одиницю.

Варіант 6.

Створити клас «трапеція», такий, що:

a) його екземпляр містить розмір двох основ і висоти a, b, h.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням a, b і h.

c) його методи дозволяють отримувати і присвоювати значення основ, висоти та обчислювати значення площі.

d) операція «<<» перевантажена: виводить на екран значення двох основ, висоти і площі трапеції.

e) операція «+» перевантажена: для двох трапецій створює нову трапецію сумарної площі (a = a1 + a2, b = b1 + b2, h = h1 + h2).

g) операція «++» збільшує і висоту і дві основи трапеції на одиницю.

Варіант 7.

Створити клас «смайлик», такий, що:

a) його екземпляр містить розмір радіуса особи R і радіуса r очей.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням R і r.

c) його методи дозволяють отримувати і присвоювати значення R і r та обчислювати значення площі.

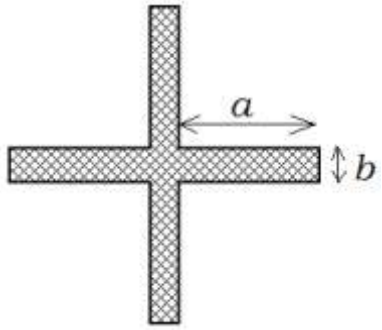
d) операція «<<» перевантажена: виводить на екран значення R, r і площі (без площі очей).

e) операція «+» перевантажена: для двох «смайликів» створює новий «смайлик» з параметрами (R = R1 + R2, r = r1 + r2).

g) операція «++» збільшує радіус «обличчя» R і радіус r очей на одиницю.

Варіант 8.

Створити клас «хрест», такий, що:



a) його екземпляр містить розміри a і b.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням a і b .

c) його методи дозволяють отримувати і присвоювати значення a і b та обчислювати значення площі.

d) операція «<<» перевантажена: виводить на екран значення a, b і площі цієї фігури.

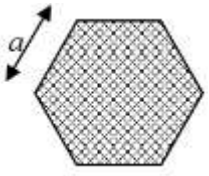
e) операція «+» перевантажена: для двох фігур створює нову фігуру з параметрами (a = a1 + a2, b = b1 + b2).

g) операція «++» збільшує a і b на одиницю.

Варіант 9.

Створити клас «шайба» (правильний шестикутник), такий, що:

$$S = \frac{3\sqrt{3}}{2}a^2$$



a) його екземпляр містить розмір сторони a.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням a.

c) його методи дозволяють отримувати і присвоювати значення a та обчислювати значення площі виконуючи при цьому відповідні перетворення.

d) операція «<<» перевантажена: виводить на екран значення сторони і площі цієї фігури.

e) операція «+» перевантажена: отримує екземпляри двох шайб і повертає нову шайбу, площа якої дорівнює сумі площ цих двох шайб. Сторона нової шайби має бути перерахована.

Варіант 10. Створити клас «ромб»:

a) його екземпляр містить значення діагоналей a і b.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням a і b.

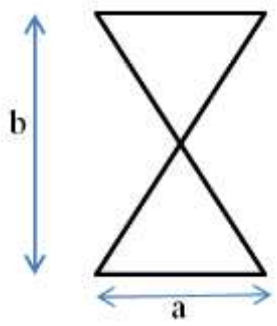
c) його методи дозволяють отримувати і присвоювати значення a і b та обчислювати значення площі.

d) операція «<<» перевантажена: виводить на екран значення діагоналей і площі цієї фігури.

e) операція «+» перевантажена: для двох фігур створює нову фігуру з параметрами (a = a1 + a2, b = b1 + b2.

g) операція «++» збільшує a і b на одиницю.

Варіант 11. Створити клас «пісочний годинник»:



a) його екземпляр містить розміри a і b.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням a і b.

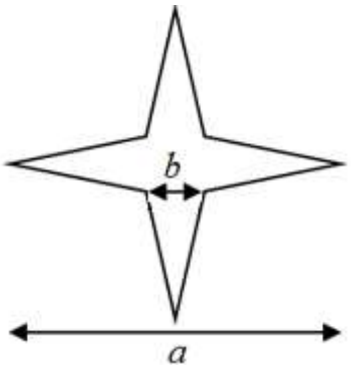
c) його методи дозволяють отримувати і присвоювати значення a і b та обчислювати значення площі.

d) операція «<<» перевантажена: виводить на екран значення a, b і площі цієї фігури.

e) операція «+» перевантажена: для двох фігур створює нову фігуру з параметрами (a = a1 + a2, b = b1 + b2).

g) операція «++» збільшує a і b на одиницю.

Варіант 12. Створити клас «зірка»:



a) його екземпляр містить розміри a і b.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням a і b.

c) його методи дозволяють отримувати і присвоювати значення a і b та обчислювати значення площі.

d) операція «<<» перевантажена: виводить на екран значення a , b і площі цієї фігури.

e) операція «+» перевантажена: для двох фігур створює нову фігуру з параметрами (a = a1 + a2, b = b1 + b2).

g) операція «++» збільшує a і b на одиницю.

Контрольні питання

1. Як можна викликати операцію-функцію?
2. Чи можна змінити кількість операндів перевантаженої операції?
3. Чи всі оператори мови С ++ можуть бути перевантажені?
4. Якими двома різними способами визначаються перевантажені операції?
5. Чи всі операції можна перевантажити за допомогою глобальної дружньої функції?
6. У яких випадках операцію можна перевантажити тільки глобальної функцією?
7. У яких випадках глобальна операція-функція повинна бути дружньою?