ЛАБОРАТОРНА РОБОТА №3

Тема: Перевантаження операцій класу

Мета: ознайомитись зі способами перевантаження операцій та навчитись використовувати їх при роботі з об'єктами.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

С++ дозволяє перевизначити дію більшості операцій таким чином, щоб при використанні з об'єктом конкретного класу вони виконували задані функції. Це дає можливість використовувати власні типи даних аналогічно до стандартних. Визначення власних операцій вводити неможливо.

Перевантаження операцій здійснюється за допомогою методів спеціального виду (функцій-операцій) і підлягає наступним правилам:

- при перевантаженні операцій зберігаються кількість аргументів, пріоритети операцій і правила асоціації (справа наліво і зліва направо), що використовуються в стандартних типах;
- для стандартних типів даних перевизначити операції неможливо;
- функції-операції не можуть мати аргументів по замовчуванню;
- функції-операції не успадковуються (за виключенням =);
- функції-операції не можуть визначатися як static.

Функцію-операцію можна визначити трьома способами:

- вона може бути методом класу;
- вона може бути дружньою функцією;
- вона може бути звичайною функцією.

У двох останніх випадках функція повинна приймати хоча б один аргумент, який має тип класу, вказівник або посилання на клас.

Функція-операція містить ключове слово operator, за яким слідує знак операції, яку потрібно перевизначити:

```
тип operator_операція (список параметрів) { тіло функції }
```

Перевантаження унарних операцій.

Унарні операції – це такі, які мають тільки один операнд (операнд – це змінна, на яку діє операція). Прикладом унарних операцій є операціїї інкремента та декремента «++» та «--», а також унарний мінус – операція зміни знаку, наприклад: x = -x;.

Для демонстрації перевантаження унарного оператора створима клас Counter, який міститиме два поля: hours ті minutes, що визначатимуть деякий час доби. У класі створено 2 конструктори: без параметрів (поточний час встановлюється рівним 0год 00хв) та параметризований, з можливістю задати певний час. Унарні оператори інкременту та декременту моделюватимуть перевід годинника на літній чи зимовий час.

Бінарні операції можуть бути перевантажені таким же чином, як і унарні. У наведеному вище коді перевантажена операція «+», а у головному коді зустрічається стрічка, де проводиться сумування двох об'єктів (c3=c1+c2). Оголошення в класі Counter виглядає наступним чином:

```
Counter operator+ (Counter);
```

Ця операція повертає значення типу Counter і приймає один аргумент того ж типу.

Для обчислення значення функції operator+ ми спочатку додаємо значення hours та minutes обох операндів (коректуючи їх в разі необхідності). Отримані значення h та m ми згодом використовуємо при ініціалізації безіменного об'єкту Counter, який буде повернутий із функції:

return Counter(h,m);

У виразі

c3=c1+c2;

важливо розуміти, до яких об'єктів будуть відноситись аргументи і значення, що повертаються. Коли компілятор зустріне цей вираз, то він переглядає типи аргументів, знайшовши тільки аргументи типу Counter, він виконає операції класу Counter operator+ (Counter);. Але який із об'єктів використовується в якості аргумента – c1 чи c2? І чи немає потреби використовувати два аргументи, оскільки ми додаєма 2 об'єкти?

Існує правило: об'єкт, що розташований зліва сторони операції (у нашому випадку c1), викликає функцію оператора. Об'єкт, що стоїть справа знака операції повинен бути переданий у функцію в якості аргумента. Операція повертає значення, яке ми згодом використовуємо для власних потреб. У функції до лівого операнда ми маємо прямий доступ, використовуючи hours та minutes, так як це об'єкт, що викликав функцію. До правого операнда ми маємо доступ як до аргумента функції, тобто t2.hours та t2.minutes.

Якщо узагальнити вищесказане, то можна сказати, що перевантаженій операції завжди потрібна кількість аргументів, на одиницю менша, ніж кількість операндів, в зв'язку з тим, що один із операндів є об'єктом, що викликає функцію. Тому для унарних операцій не потрібні аргументи (за винятком функцій і операторів, які є дружніми для класу).

ПРИКЛАД ПРОГРАМИ

```
/* Створити клас комплексні числа.
Визначити необхідні конструктори та деструктор.
Перевантажити потокові операції введення і виведення,
операції + , - , * , / та ^ .
Обчислити значення виразу y=a*x2+b*x+c для комплексних коефіцієнтів
 a, b, c у комплексній точці x.*/

#include <iostream>
#include <assert.h>

using namespace std;

class Complex
{
	double re;
	double im;
public:
	Complex(double re = 0.0, double im = 0.0);
	~Complex();
	Complex operator+(Complex&);
	Complex operator-(Complex&);
	Complex operator*(Complex&);
```

```
	Complex operator/(Complex&);
	Complex operator^(unsigned);
	friend istream& operator>>(istream&,Complex&);
	friend ostream& operator<<(ostream&,Complex&);
};

Complex::Complex(double re, double im)
{
	this->re=re;
	this->im=im;
}

Complex::~Complex()
{
}

Complex Complex::operator+(Complex& y)
{
	return Complex(re+y.re, im+y.im);
}

Complex Complex::operator-(Complex& y)
{
	return Complex(re-y.re, im-y.im);
}

Complex Complex::operator*(Complex& y)
{
	return Complex(re*y.re-im*y.im, re*y.im+im*y.re);
}

Complex Complex::operator/(Complex& y)
{
	double r1=re;
	double i1=im;
	double r2=y.re;
	double i2=y.im;
	return Complex((r1*r2-i1*i2)/(r2*r2+i2*i2),(-
r1*i2+i1*r2)/(r2*r2+i2*i2));
}

Complex Complex::operator^(unsigned n)
{
	Complex y(1,0);
	for(unsigned i=1;i<=n;i++)
		y=y*(*this);
	return y;
}

istream& operator >>(istream& is, Complex& x)
{
	char c;
	is>>x.re;
	cin>>c;
	assert(c==',');
```

```
        is>>x.im;
        return is;
}

ostream& operator <<(ostream& os, Complex& x)
{
        os<<x.re<<','<<x.im<<endl;
        return os;
}

int main()
{
        Complex a(1,1);
        Complex b(1,1);
        Complex c(1,1);
        Complex x;
        cout << "Введіть комплексне число у форматі: re,im ";
        cin >> x;
        cout << "Результат = " << a*(x^2)+b*x+c << endl;
        return 0;
}
```

ПОРЯДОК ВИКОНАННЯ РОБОТИ

Завдання 1. В класі Integer, який розроблений в завданні №1 лабораторної роботи №1, перевизначте чотири цілочисельні арифметичні операції («+», «-», «*» , «/») так, щоб їх можна було використовувати для операцій з об'єктами класу Integer. Перевірте роботу перевизначених операцій за допомогою функції main, яка виглядає наступним чином:

```
int main(){
    Integer objA(10);
    Integer objB(x);   // де x - номер у списку групи
    Integer objC;
    objC = objA + objB;
    objC.print();
    objC = objA - objB;
    objC.print();
    objC = objA * objB;
    objC.print();
    objC = objA / objB;
    objC.print();
}
```

Якщо результат будь-якої з операцій виходить за межі типу даних int, що може мати значення від 2 147 483 648 до -2 147 483 648, то операція повинна послати повідомлення про помилку і завершити програму. Такі типи даних корисні там, де помилки можуть бути викликані арифметичним переповненням, що неприпустимо.

Підказка: для полегшення перевірки переповнення виконуйте обчислення з використанням типу даних long.

Завдання 2. Для розробленого у лабораторній роботі № 2 класу перевизначити операції:

- зчитування з потоку вводу sdt::cin: `sdt::cin >> myObject;`
- виводу у потік std::cout: `std::cout << myObject`.

ОФОРМЛЕННЯ ЗВІТУ:

1. Титульний лист.
2. Мета роботи.
3. Текст завдання згідно варіанту.
4. Структурна UML-діаграма класу.
5. Код програми.
6. Screen-shot вікна виконання програми.
7. Висновки.

Контрольні питання

1. Що таке механізм перевантаження оператора?

2. Назвіть причини необхідності перевантаження операторів?

3. Як можна перевантажити оператор для класу?

5. Які оператори перевантажувати не можна?

6. Яка різниця між перевантаженням бінарних та унарних операторів?

7. Опишіть синтаксис перевантаження унарних операторів класу.

8. У чому полягає специфіка перевантаження операторів зсуву

9. У яких випадках перевантаження операцій визначається як дружня функція до класу?