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 слідє знак перевизначеної операції:

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, наприклад -35.

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 використовуємопри ініціалізації безіменного об’єкту Counter, який буде повертати значення:

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, im;
public:
	complex(double=0,double=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 r, double i)
{
	re=r; im=i;
}

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. Ознайомитись зі способами перевантаження операцій у С++.

Завдання 2. Проаналізувати приклад програми, яка оголошує клас та перевантажує низку операцій над точками у тривимірному просторі. Визначити результат її роботи.

Завдання 3. В класі Int, 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 Int. Якщо результат будь-якої з операцій виходить за межі типу int (в 32-бітній системі), 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, яке неприпустимо.

Напишіть програму для перевірки цього класу.

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

Завдання 4. Для заданого варіанта індивідуального завдання виконати перевантаження операцій для зручності роботи з об’єктами. При необхідності оголосіть певні операторні функції друзями класу.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ

1. Створити клас – вектор, який має у закритій частині вказівник на дані цілого типу та кількість елементів. Визначити необхідні конструктори, деструктор. Перевантажити потокові операції введення і виведення елементів вектора, операції + , – , * , = та [ ] .

2. Створити клас – матриця, який у закритій частині містить вказівник на дані цілого типу, кількість рядків і стовпців. Визначити необхідні конструктори,

деструктор. Перевантажити потокові операції введення і виведення елементів матриці, операції + , – , * та = .

3. Створити клас – дата з полями у закритій частині: день (1-31), місяць (1-12), рік (ціле число). Визначити необхідні конструктори, методи доступу, деструктор. Перевантажити потокові операції введення і виведення, операції + (збільшення на n днів) , – (різниця між двома датами), < та >= .

4. Створити клас – час з полями у закритій частині: година (0-23), хвилини (0-59), секунди (0-59). Визначити необхідні конструктори, методи доступу, деструктор. Перевантажити потокові операції введення і виведення, операції += (збільшення на n секунд), – (різниця між двома часами), <= та > .

5. Створити клас – прямокутник. У закритій частині описати поля – висоту і ширину. Визначити необхідні конструктори, методи доступу, деструктор. Перевантажити потокові операції введення і виведення, операції порівняння за площею < , > та == .

6. Створити клас – велике ціле число, яке не може бути зображене одним значенням вбудованих типів і умовно може бути утворене з декількох таких значень. У закритій частині визначити поля – вказівник на дані цілого типу та кількість елементів. Визначити необхідні конструктори, деструктор. Перевантажити потокові операції введення і виведення, операції + , – , < , > та == .

7. Створити клас – двозв’язний список. Визначити необхідні конструктори, деструктор. Перевантажити потокові операції введення і виведення, операції + (додавання елемента в початок списку), += (додавання елемента в кінець списку), – (вилучення заданого елемента зі списку), ++ (перехід до наступного елемента), -- (перехід до попереднього елемента).

8. Створити клас – однозв’язний список. Визначити необхідні конструктори, деструктор. Перевантажити потокові операції введення і виведення, операції + (додавання елемента в початок списку), += (додавання елемента в кінець списку), – (вилучення заданого елемента зі списку), ++ (перехід до наступного елемента), -- (перехід до попереднього елемента), [ ] (одержання елемента списку).

9. Створити клас – коло. У закритій частині описати поля – координати центру та радіус. Визначити необхідні конструктори, методи доступу, деструктор. Перевантажити потокові операції введення і виведення, операції + (збільшення радіусу), – (зменшення радіусу), порівняння за площею < , > та == .

10. Створити клас – ціле число. У закритій частині визначити поля – система числення і рядок символів, що відповідає числу. Визначити необхідні конструктори, методи доступу, деструктор. Перевантажити потокові операції введення і виведення, вважаючи що слід вводити десяткові числа і систему числення, а виводити – число у обраній системі.

11. Створити клас – квадрат з полями у закритій частині: координати головної діагоналі. Визначити необхідні конструктори, методи доступу, деструктор. Перевантажити потокові операції введення і виведення, операції + (збільшення головної діагоналі), – (зменшення головної діагоналі), порівняння за периметром < , > та == .

12. Створити клас – стек. Визначити необхідні конструктори, деструктор. Перевантажити потокові операції введення і виведення, операції + (додавання елемента у стек), – (вилучення елемента зі стеку), == та != .

13. Створити клас – рядок символів. У закритій частині описати вказівник на символьний тип (на початок рядка). Визначити необхідні конструктори, деструктор. Перевантажити потокові операції введення і виведення, операції + (конкатенації рядків), * (пошуку входження підрядка у рядок), – (вилучення підрядка з рядка).

14. Створити клас – квадратна матриця. У закритій частині описати поля: розмір матриці та вказівник на її початок в області динамічної пам’яті. Визначити необхідні конструктори, деструктор. Перевантажити потокові операції введення і виведення, порівняння за слідом < , > , == та !=.

15. Створити клас – множина цілих чисел. У закритій частині описати вказівник на цілий тип (на елементи множини). Визначити необхідні конструктори, деструктор. Перевантажити потокові операції введення і виведення елементів множини, + (об’єднання), – (різниці) та * (перетину) множин.

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?