Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



Звіт

З лабораторної роботи №5

З дисципліни «Об’єктно-орієнтоване програмування »

На тему:

ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ

Виконав: Куцевол В.С. KI-109

Прийняв: асис. каф. ЕОМ

Гузинець Н.В

Львів – 2024

**Тема роботи:** ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ

**Мета:** познайомитися із перевантаженням операторів.

**Теоретичні відомості**

Щоб використовувати операцію над об'єктами класів, ця операція повинна бути перевантажена, але є два виключення. Операції присвоювання (=) і взяття адреси (&) створюються в класі автоматично за замовчуванням, тому їх можна використовувати без явного перевантаження. За замовчуванням операція присвоювання зводиться до побітового копіювання даних-елементів класу. Проте таке побітове копіювання небезпечне для класів з елементами, що вказують на динамічно виділені області пам'яті, масиви, рядки, оскільки в цьому випадку відбувається копіювання не даних (глибоке копіювання), а лише вказівників на дані (поверхневе копіювання). Для таких класів слід явно перевантажувати операцію присвоювання і здійснювати у ній глибоке копіювання. Операція адресації також може бути використана з об'єктами будь-яких класів без перевантаження. Вона просто повертає адресу об'єкта в пам'яті. Але операцію адресації можна також і перевантажувати.

Перевантаження операцій підпорядковується наступним правилам:

• При перевантаженні зберігаються кількість аргументів, пріоритети операцій та правила асоціації, що використовуються у стандартних типах даних;

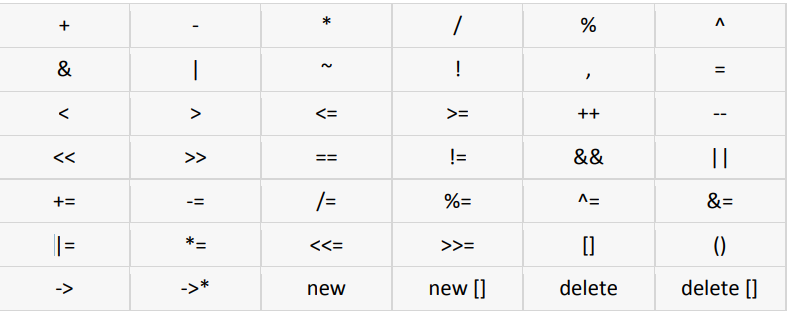
• Для стандартних типів даних операції не підлягають перевизначенню; 78

• Перевантажена функція-оператор не може мати параметрів по замовчуванню, не успадковується та не може бути визначеною як static;

• Функція-оператор може бути визначена трьома способами – метод класу, дружня функція або звичайна функція. В останніх двох випадках вона повинна приймати хоча б один аргумент, що має тип класу, вказівника або посилання на клас.

При перевантаженні операцій ( ), [], -> та = функція перевантаження операції може бути оголошена лише як метод класу. Для інших операцій функції перевантаження операцій можуть не бути методами класу.

Оператори, які можна перевантажити:



Оператори, які не можна перевантажити:

- sizeof

- . (селектор елемента структури або класу)

- \* (оператор доступу до елементу за вказівником)

- :: (оператор дозволу видимості)

- ?: (тернарний оператор)

- typeid

- const\_cast

- dynamic\_cast

- reinterpret\_cast

- static\_cast

- # і ## (символи препроцесору)

Коли операція реалізована як метод класу, тоді: 79 - якщо операція є унарною (передбачає один операнд, наприклад, інкременту або []), то лівим операндом вважаєтсья об‘єкт, до якого застосовується операція, а правий операнд відсутній, тому метод, що реалізує даний оператор не приймає аргументів, за виключенням кількох операторів (наприклад, постфіксна форма інкременту або декременту).

- якщо операція є бінарною (передбачає 2 операнди, наприклад додавання або присвоєння) і лівий операнд є об‘єктом класу у якому перевантажується операція, то крайній лівий операнд вважається об‘єктом зпід якого здійснюється виклик даного методу (оператора), а правий операнд – передається як параметр, тому для нього слід вказати тип аргумента у методі; перевантажених операторів одного виду може бути кілька в залежності від типу аргументу, що передається методу;

Якщо операція є бінарною але лівий операнд не є об‘єктом класу у якому перевантажується операція, то дана операція не може бути методом класу, а має бути реалізована як дружня функція, якщо ця функція повинна мати прямий доступ до закритих або захищених елементів цього класу, або звичайна функція в протилежному випадку.

Перевантажена операція << повинна мати лівий операнд типу ostream & (такий, як cout), так що вона не може бути функцією-елементом. Аналогічно, перевантажена операція >> повинна мати лівий операнд типу іstream & (такий, як cіn), так що вона теж не може бути функцією-елементом. До того ж кожна з цих перевантажених функцій-операцій може забажати доступу до закритих елементів-даних об'єкта класу, так що ці перевантажені функції-операції роблять функціями-друзями класу

Будь-яку бінарну операцію можна перевантажувати як нестатичний метод з одним аргументом, або як функцію, що не є елементом, із двома аргументами (один з цих аргументів повинен бути або об'єктом класу, або посиланням на об'єкт класу).

Унарну операцію класу можна перевантажувати як метод без аргументів, або як функцію, з одним аргументом; цей аргумент повинен бути або об'єктом класу, або посиланням на об'єкт класу. Функції-елементи, що реалізують перевантажені операції, повинні бути нестатичними, щоб вони могли мати доступ до даних класу.

При перевантаженні унарних операцій переважно створюють методи класу, замість дружніх функцій, що не є членами класу. Дружніх функцій краще уникати доти, поки вони не стануть абсолютно необхідними. Використання друзів порушує інкапсуляцію класу

Щоб перевантажити операцію інкремента та декремента для одержання можливості використання і префіксної, і постфіксної форм, кожна з цих двох перевантажених функцій-операцій повинна мати різну сигнатуру, щоб компілятор мав можливість визначити, яка версія мається на увазі в кожному конкретному випадку. Префіксний варіант перевантажується як будь-яка інша префіксна унарна операція. Для постфіксної форми вводиться додатковий параметр цілого типу у список аргументів, щоб зробити функцію для постфіксного варіанту відмінною від функції для префіксної форми

Зауваження щодо перевантаження операцій:

• Неможливим є введення власних операторів.

• Компілятор С++ не розуміє семантики перевантаженого оператору, а отже, не нав'язує жодних математичних концепцій. Можна перевантажити, скажімо, оператор інкременту в якості зменшення аргументу, проте навряд чи в цьому є сенс.

• Не існує виведення складних операторів з простих: якщо ви перевантажили оператори operator+ та operator=, це зовсім не означає, що С++ обчислить вираз a += b, оскільки ви не перевантажили operator +=.

• Перевантаження бінарних операторів не тотожньо відносно перестановки аргументів місцями, тим більше, якщо вони різного типу.

Синтаксис виклику операторів реалізований у мові С++ таким чином, щоб програміст міг записати операцію над об‘єктом класу з звичному для нього вигляді, наприклад object1 += object2. Проте для компілятора такий запис стосовно об‘єктів не є звичним, бо дана операція у такому вигляді визначена лише для простих типів, а для об‘єкту класу даний виклик має бути перетворений у виклик методу, що оголошений в класі, або у виклик відповідної функції.

Дріжніми до класу можуть бути: класи, методи та функції. Якщо якийсь з даних елементів має бути дружнім до класу, от в цьому класі його слід оголосити як дружній. Для цьо слід зробити: - У випадку якщо дружнім має бути клас, то цей клас треба оголосити дружнім в тілі класу, до якого він має бути дружнім за допомогою ключового слова friend

У цьому випадку всі елементи дружнього класу матимуть доступ до закритих і захищених елементів класу, до якого вони є дружніми. Оскільки тут someClass ідентифікується за допомогою ключового слова class як клас, то випереджуючого оголошення для компілятора робити не потрібно.

У цьому випадку лише вказаний метод класу матиме доступ до закритих і захищених елементів класу, до якого він є дружнім. Оскільки з одної сторони класи з дружніми методами переважно оголошуються після класів, до яких вони мають бути дружніми, але з іншої сторони клас, що містить дружній метод, має бути відомим до моменту оголошення методу дружнім, то перед оголошенням класу, до якого метод класу має бути дружніми слід вжити випереджаюче оголошення класу, що містить дружній метод. Інакше компілятор видасть помилку, бо для нього ім‘я класу, що міститить дружній метод буде невідомим.

**Хід виконання роботи**

**Варіант 12**

Завдання : Розширити функціональність розроблених у 4 лабораторній роботі класів за допомогою операторів, що задані варіантом та оператора присвоювання. Конкретні функції операторів реалізувати на власний розсуд (крім оператора присвоювання). Організувати виведення та введення даних за допомогою класів-потоків сin, cout та перевантажених операторів вводу/виводу. Написати програму, яка демонструє роботу з об'єктами цього класу.



//file CBorwser.h

#pragma once

#include <iostream>

#include <string>

using std::string;

using std::ostream;

using std::istream;

using std::endl;

class CBrowser

{

private:

string URL;

int pageSize;

unsigned int upload;

unsigned int download;

public:

int GetPageSize();

int GetUpload();

string GetURL();

int GetDownload();

void SetPageSize(int);

void SetUpload(int);

void SetURL(const string&);

void SetDownload(int);

float UploadFile(int filesize, int internetSpeed);

float DownloadFile(int filesize, int internetSpeed);

CBrowser(string URL, int pageSize, unsigned int upload, unsigned int dowload);

CBrowser();

friend CBrowser& operator += (CBrowser& obj1, CBrowser& obj2);

friend CBrowser& operator -= (CBrowser& obj1, CBrowser& obj2);

friend bool operator < (CBrowser obj1, CBrowser obj2);

friend bool operator >= (CBrowser obj1, CBrowser obj2);

friend bool operator > (CBrowser obj1, CBrowser obj2);

friend bool operator <= (CBrowser obj1, CBrowser obj2);

friend ostream& operator << (ostream& cout, CBrowser obj);

friend istream& operator >> (istream &cin, CBrowser &obj);

};

//file CBrowser.cpp

#include "CBrowser.h"

#include<iostream>

#include<string>

using std::string;

using std::cin;

using std::cout;

using std::endl;

int CBrowser::GetUpload() {

return CBrowser::upload;

}

int CBrowser::GetDownload() {

return CBrowser::download;

}

int CBrowser::GetPageSize() {

return CBrowser::pageSize;

}

string CBrowser::GetURL() {

return this->URL;

}

void CBrowser::SetDownload(int download) {

CBrowser::download = download;

}

void CBrowser::SetUpload(int upload) {

CBrowser::upload = upload;

}

void CBrowser::SetPageSize(int pageSize) {

CBrowser::pageSize = pageSize;

}

void CBrowser::SetURL(const string& URL) {

CBrowser::URL = URL;

}

float CBrowser::DownloadFile(int filesize, int internetSpeed) {

return filesize / internetSpeed;

}

float CBrowser::UploadFile(int filesize, int internetSpeed) {

return filesize / internetSpeed;

}

CBrowser::CBrowser(string URL, int pageSize, unsigned int upload, unsigned int dowload) {

CBrowser::URL = URL;

CBrowser::pageSize = pageSize;

CBrowser::download = download;

CBrowser::upload = upload;

}

CBrowser::CBrowser() {

CBrowser::URL = "https://www.youtube.com/watch?v=dQw4w9WgXcQ";

CBrowser::pageSize = 1;

CBrowser::download = 1;

CBrowser::upload = 1;

}

CBrowser& operator += (CBrowser& obj1, CBrowser& obj2) {

int newPageSize = obj1.pageSize + obj2.pageSize;

obj1.SetPageSize(newPageSize);

return obj1;

}

CBrowser& operator -= (CBrowser& obj1, CBrowser& obj2) {

int newPageSize = obj1.pageSize -= obj2.pageSize;

obj1.SetPageSize(newPageSize);

return obj1;

}

bool operator < (CBrowser obj1, CBrowser obj2) {

return obj1.pageSize < obj2.pageSize;

}

bool operator >= (CBrowser obj1, CBrowser obj2) {

return !(obj1 < obj2);

}

bool operator > (CBrowser obj1, CBrowser obj2) {

return obj1.pageSize > obj2.pageSize;

}

bool operator <= (CBrowser obj1, CBrowser obj2) {

return !(obj1 > obj2);

}

ostream& operator << (ostream& cout, CBrowser obj) {

cout << "URL:" << obj.URL << endl

<< "Download: " << obj.download << endl

<< "Upload: " << obj.upload << endl

<< "Page size: " << obj.pageSize << endl;

return cout;

}

istream& operator >> (istream &cin, CBrowser &obj) {

string URL;

int pageSize;

unsigned int upload;

unsigned int download;

cout << "Enter new URL: "; cin >> URL;

cout << "Enter new page size: "; cin >> pageSize;

cout << "Enter new upload: "; cin >> upload;

cout << "Enter new download: "; cin >> download;

obj.SetURL(URL);

obj.SetPageSize(pageSize);

obj.SetUpload(upload);

obj.SetDownload(download);

return cin;

}

//file main.cpp

#include <iostream>

#include "CBrowser.h"

using std::string;

using std::cin;

using std::cout;

using std::endl;

int main() {

int internetSpeed, filesize;

cout << "Enter your internet speed (Mb/s): ";

cin >> internetSpeed;

cout << "Enter size of your file(Mb): ";

cin >> filesize;

CBrowser base = CBrowser();

cout << "Base URL is: " << base.GetURL() << endl;

cout << "Base download is: " << base.GetDownload() << endl;

cout << "Base upload is: " << base.GetUpload() << endl;

cout << "Base page size is: " << base.GetPageSize() << endl;

CBrowser chrome = CBrowser();

chrome.SetDownload(100);

chrome.SetPageSize(50);

chrome.SetUpload(40);

chrome.SetURL("https://www.youtube.com/");

cout << "-------------------------------------------" << endl;

cout << "Chrome URL is: " << chrome.GetURL() << endl;

cout << "Chrome download is: " << chrome.GetDownload() << endl;

cout << "Chrome upload is: " << chrome.GetUpload() << endl;

cout << "Chrome page size is: " << chrome.GetPageSize() << endl;

cout << "Chrome will upload your file in: " << chrome.UploadFile(filesize, internetSpeed) << endl;

cout << "Chrome will download your file in: " << chrome.DownloadFile(filesize, internetSpeed) << endl;

cout << "Lab 5 realization:" << endl;

base += chrome;

cout << "Base`s page size has been changed by operator (+=): " << base.GetPageSize()<< endl;

base -= chrome;

cout << "Base`s page size has been changed by operator (-=) : " << base.GetPageSize() << endl;

if (base > chrome) {

cout << "Base`s page size is bigger then chrome`s" << endl;

}

if (base < chrome) {

cout << "Base`s page size is smaller then chrome`s" << endl;

}

cin >> chrome;

cout << chrome;

return 0;

}

**Висновок**

У ході цієї лабораторної роботи я поглибив своє розуміння перевантаження операторів у мові програмування C++. Перевантаження операторів дозволяє створювати спеціальні функції, які виконуються при використанні операторів з користувацькою структурою даних або об'єктами класів.

Під час цієї роботи я вивчив, як перевантажувати різні оператори, такі як арифметичні, логічні та присвоювання, для власних класів та структур. Я також досліджував можливості перегрузки операторів для створення більш зрозумілого та ефективного коду.

Ця лабораторна робота допомогла мені краще розібратися з технікою перевантаження операторів та її застосуванням у практичних завданнях. Я впевнений, що отримані знання і навички стануть корисними для мене у подальшій роботі з програмуванням на C++, дозволяючи створювати більш ефективні та зрозумілі програми.

Top of Form