Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



Звіт

З лабораторної роботи №4

З дисципліни «Об’єктно-орієнтоване програмування »

На тему:

КЛАСИ ТА ОБ’ЄКТИ

Виконав: Куцевол В.С. KI-109

Прийняв: асис. каф. ЕОМ

Гузинець Н.В

Львів – 2024

**Тема роботи:** Класи та об’єкти

**Мета:** познайомитися із класами та об‘єктами.

**Теоретичні відомості**

Основна відмінність будь-якої об‘єктно-орієнтованої мови програмування від інших не об‘єктно-орієнтованих мов програмування є можливість створення нових визначених користувачем типів, що називаються класами. Клас – це визначений користувачем тип з даними-елементами (властивостями) та функціями-елементами (методами), що являються членами класу. Він зазвичай описує певну абстракцію якоїсь сутності реального світу з її властивостями і можливими діями. Змінна типу клас називається об‘єктом. Об‘єкт – це вже не абстракція, а елемент реального світу, який може володіти певними характеристиками, які задаються властивостями в класі, та діяти згідно операцій заданих методами у класі. Оголошення класу в мові С++ має наступний синтаксис (не обов‘язково щоб клас мав всі секції чи спадкував базовий клас):

Члени класу (властивості і методи), оголошені після ключового слова publіc стають відкритими членами. Це означає, що вони доступні

• усім іншим членам класу;

• дружнім конструкціям (класам, методам, функціям);

• членам похідних класів;

• з-під об‘єктів класу після створення об‘єктів; після створення об‘єкту класу до його членів можна звертатися лише якщо вони є загальнодоступними.

Члени класу, оголошені після ключового слова prіvate, стають закритими членами. Вони доступні:

• для інших членів того ж класу;

• друзям класу.

Якщо в класі не оголошено секцій, але визначено методи чи властивості, то вони вважатимуться такими, що оголошені у секції prіvate. Для модифікації властивостей, що оголошені в секції private створюються спеціальні методи, які здійснюють цю модифікацію. Назви таких методів прийнято починати зі слова set. На жаргоні такі методи інколи називають сетерами. Для читання властивостей, що оголошені в секції private створюються спеціальні методи, які безпечно повертають значення цих властивостей. Назви таких методів прийнято починати зі слова get. На жаргоні такі методи інколи називають ґетерами.

Члени класу, оголошені після ключового слова protected, стають захищеними членами.

Оголошення класу містить оголошення даних-елементів (властивостей) та функцій-елементів (методів) класу. Одна з фундаментальних основ ООП передбачає інкапсуляцію даних, тобто дані мають бути недоступними ззовні, а 49 лише за посередництвом загальнодоступних методів класу. Оголошення методів має вигляд оголошення прототипу функції в середині однієї з секцій класу, зазвичай у секції public. Якщо методи є службовими і не мають бути доступні ззовні, тоді їх розміщують в секції private або protected. Сама ж реалізація методу може знаходитись як усередині класу (тоді оголошення методу в класі має вигляд оголошення функції з тілом), так і поза оголошенням класу (тоді у класі оголошується прототип функції, а її тіло визначається за межами класу). Але один з найфундаментальніших принципів розробки програмного забезпечення мовою С++ полягає у розмежуванні інтерфейсу класу від його реалізації. Тому при побудові програми мовою C++ кожне оголошення класу зазвичай розміщують у заголовочний файл \*.h назва якого співпадає з назвою класу, а реалізацію методів цього класу – у файл \*.cpp із тим ж іменем, що й \*.h файл. Заголовочні файли включаються (за допомогою директиви #іnclude) у кожен файл, у якому використовується клас, а файли з вихідними кодами компілюються і компонуються з файлом, що містить головну програму (main-функцію).

При визначенні методу класу за межами класу необхідно зв'язати ім'я відповідного методу з ім'ям класу. Зв'язок встановлюється шляхом написання імені класу, оператора розширення області видимості (::) та імені методу. Оскільки різні класи можуть мати елементи з однаковими іменами, то оператор розширення області видимості дозволяє однозначно ідентифікувати методи конкретного класу.

Областю видимості властивостей та методів класу є клас, тобто все що оголошене в середині класу є видиме і доступне за іменем елементів в середині класу без додаткових маніпуляцій.

Статичні дані-елементи (властивості) оголошуються в класі, а ініціалізуються за межами класу (не у конструкторах). На відміну від інших властивостей вони належать класу, а не об‘єкту. Тобто вони єдині для всіх об‘єктів певного класу і зміни одним об‘єктом вплинуть на всі об‘єкти даного 50 класу. Якщо статичні властивості оголошені в секції public, то вони будуть завжди доступні незалежно від того чи створено хоч один об‘єкт класу. За допомогою статичних властивостей зручно реалізовувати лічильники, наприклад, лічильники створених об‘єктів класу.

Статичні методи оголошуються з використанням ключового слова static при оголошенні методу. На відміну від нестатичних методів дані методи існують завжди, навіть якщо нема жодного екземпляру класу. Тому їх можна викликати з будь якого місця програми не створюючи при цьому об‘єкту класу. Для виклику статичного методу слід вказати назву класу, оператор розширення області видимості (::), назву статичного методу, передати йому параметри і поставити крапку з комою в кінці виклику. Статичні методи не мають доступу до членів класу, тому вони мають реалізуватися самодостатніми, тобто такими, що не потребують для своєї роботи інших даних, ніж ті що вони містять в собі і ті, що передаються їм через параметри. Одним з прикладів використання статичних методів є методи, що обчислюють різні тригонометричні функції

При створенні об'єкта класу автоматично викликається спеціальний метод, що зветься конструктор. Конструктор – це метод класу ім‘я якого співпадає з іменем класу та не повертає ніяких значень (навіть void). Основне призначення конструктора – початкова ініціалізація об‘єкту класу

У класі може бути визначено кілька перевантажених конструкторів, що відрізняються списком параметрів. При створенні об'єкта викликатися буде тільки один з них. Який саме – визначається зі способу створення об'єкта. Якщо конструктор має один параметр, то він може бути використаним для операції приведення типів. Якщо треба унеможливити використання конструктора для операції приведення типів, то його слід оголосити з використанням ключового слова explicit.

Конструктор без параметрів або конструктор, у якого всі аргументи – це аргументи за замовчуванням, має спеціальну назву – конструктор за замовчуванням. Конструктор за замовчуванням може бути лише один. Якщо в класі явно не визначений конструктор за замовчуванням, то він створюється автоматично і при виклику ініціалізуватиме всі члени-дані нулями.

Конструктор, що як свій єдиний параметр приймає посилання на константний об'єкт цього ж класу, зветься конструктором копіювання. Він призначений для створення копії існуючого об'єкта шляхом поелементного копіювання значень нестатичних членів-даних (властивостей) класу і викликається при ініціалізації в операторі оголошення об'єкта або при передачі об'єкта класу за значенням у функцію (метод). Якщо конструктор копіювання не визначений явно в класі, то він створюється автоматично і при виклику здійснює поелементне копіювання всіх властивостей класу. Даний конструктор можна не оголошувати явно в більшості випадків. Одним з випадків коли його слід обов‘язково явно оголосити є ситуація коли клас містить динамічні членидані. У цьому випадку якщо не оголосити явно конструктор копіювання при копіюванні динамічних властивостей створюватимуться не їх копії, а копії вказівників на одне і те ж значення, що належить об‘єкту оригіналу. Це призведе до помилок функціонування програми, зокрема спроби більше, ніж один раз знищити динамічно виділену властивість при знищенні об‘єктів оригіналу і його копій.

Деструктор – це спеціальна функція-елемент класу, яка викликається при знищені об‘єкту і не приймає параметрів та не повертає значень. Знищення об'єкту може відбуватися, наприклад, коли виконувана програма залишає область дії, у якій був створений об'єкт цього класу, або явно здійснюється знищення динамічно створеного об‘єкту. Деструктор сам не знищує об'єкт, а лише виконує підготовку до знищення об‘єкту (вивільняє пам‘ять від динамічно створених даних-елементів об‘єкту, закриває потоки обміну даними,…) перед тим, як система звільняє область пам'яті, у якій зберігався об'єкт, щоб використовувати її для розміщення нових об'єктів. Ім'я деструктора складається з символ тильда (~) та імені класу. Клас може мати тільки один деструктор – перевантаження деструктора забороняється. Якщо деструктор не визначений явно, то він буде згенерований автоматично і при виклику нічого не робитиме. Деструктор доцільно оголошувати написавши перед ним ключове слово virtual, яке забезпечує коректність функціонування класу при спадкуванні.

Загалом у класі автоматично визначаються наступні методи:

- конструктор за замовчуванням;

- конструктор копіювання; 54

- оператор присвоювання;

- деструктор;

- оператор адресації.

Конструктори глобальних об'єктів (ті що створюються за межами будьякої функції) викликаються перед тим як будь-яка функція даного файлу (включаючи maіn) почне виконуватися. Відповідні деструктори викликаються, коли завершується maіn-функція або коли викликається функція exіt.

Конструктори автоматичних локальних об'єктів викликаються, коли процес виконання досягає місця де створюються об'єкти класів. Відповідні деструктори викликаються, коли покидається область дії об'єктів (тобто покидається блок, у якому ці об'єкти оголошені). Конструктори і деструктори для автоматичних об'єктів викликаються щораз при вході і виході з області дії.

Конструктори статичних локальних об'єктів викликаються відразу ж, як тільки процес виконання досягає місця, де об'єкти були вперше оголошені. Відповідні деструктори викликаються, коли завершується maіn-функція або коли викликається функція exіt

**Хід виконання роботи**

**Варіант 12**

Завдання : Спроектувати клас, що реалізує предметну область згідно варіанту, та розробити програму-драйвер (main-функцію), яка демонструє роботу класу, а саме: можливі способи створення об‘єктів класів і коректність функціонування розроблених методів класу (запустити кожен метод класу хоча б 2 рази)

12. Клас CBrowser (Інтернет-браузер). Клас зберігає адресу поточної відкритої сторінки, розмір сторінки, швидкість під‘єднання до інтернету (завантаження і відвантаження). Конструктор має ініціалізувати ці властивості. Методи дозволяють модифікувати і читати властивості, завантажувати і відвантажувати файли з і на сторінку повертаючи при цьому час обміну інформації, виводити на екран поточний стан об‘єкта

//file CBrowser.h

#pragma once

#include <string>

using std::string;

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();

};

//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;

}

//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;

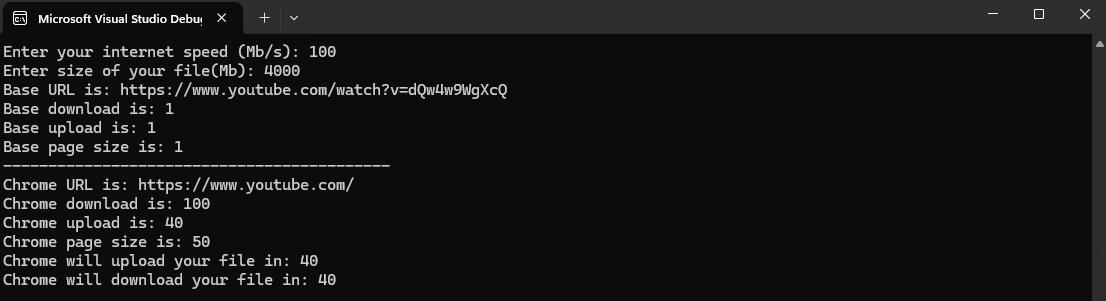
cout << "Chrome page size is: " << chrome.GetPageSize() << endl;

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

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

return 0;

}



**Висновок**

У ході цієї лабораторної роботи я мав можливість ознайомитися з основами класів та об'єктів в мові програмування C++. Робота з класами та об'єктами є фундаментальною складовою розробки програм і важливою для розуміння об'єктно-орієнтованого програмування.

Під час виконання роботи я зрозумів, що класи - це зручний спосіб організації коду, коли ми маємо справу з об'єктами, що мають спільні властивості та функціональність. Створення класів дозволяє нам визначити властивості та методи, які реалізують певний функціонал, а об'єкти цих класів дозволяють нам створювати конкретні екземпляри цих визначених об'єктів.

Вивчення цієї теми надало мені можливість краще розуміти концепції інкапсуляції, спадкування та поліморфізму. Ці концепції відіграють ключову роль у створенні складних програмних рішень, які легко підтримувати та розширювати.

Завдяки цій лабораторній роботі я набув нові знання та навички, які безумовно стануть у пригоді у моїй подальшій роботі як програміста.

Top of Form