Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**Звіт з лабораторної роботи №4**

на тему:

“Класи та об’єкти”

**Виконав:**

Ст. гр. КІ-15

Залевський К.С.

**Перевірив:**

Викладач

Козак Н.Б.

Львів – 2019

**Мета роботи:** 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.



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. Вони доступні:

· для інших членів того ж класу;

· членам похідних класів;

· друзям класу.

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 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.

**Конструктори та деструктор**

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 члени-дані. 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.

***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).



*Код програми:*

#include <iostream>

#include <string>

using namespace std;

class CComputer

{

private:

string cpumodel, videomodel;

double operation\_per\_seconds;//"operation per second"

int hardsize, ramsize, optime, numprogram;

public:

CComputer(string Vcpumodel, string Vvideomodel, double Voperation\_per\_seconds, int Vhardsize, int Vramsize, int Voptime, int Vnumprogram) //Конструктор

{

cpumodel = Vcpumodel;

videomodel = Vvideomodel;

operation\_per\_seconds = Voperation\_per\_seconds;

hardsize = Vhardsize;

ramsize = Vramsize;

optime = Voptime;

numprogram = Vnumprogram;

}

//Setters

void Set\_cpumodel(string Scpumodel)

{

cpumodel = Scpumodel;

}

void Set\_videomodel(string Svideomodel)

{

videomodel = Svideomodel;

}

void Set\_operation\_per\_seconds(double Soperation\_per\_seconds)

{

operation\_per\_seconds = Soperation\_per\_seconds;

}

void Set\_hardsize(int Shardsize)

{

hardsize = Shardsize;

}

void Set\_ramsize(int Sramsize)

{

ramsize = Sramsize;

}

void Set\_optime(int Soptime)

{

optime = Soptime;

}

//Getters

string Get\_cpumodel()

{

return cpumodel;

}

string Get\_videomodel()

{

return videomodel;

}

double Get\_operation\_per\_seconds()

{

return operation\_per\_seconds;

}

int Get\_hardsize()

{

return hardsize;

}

int Get\_ramsize()

{

return ramsize;

}

int Get\_optime()

{

return optime;

}

int Get\_numprogram()

{

return numprogram;

}

//Install and uninstall programm

void install()

{

cout << endl << "Enter the name of the application which you want to install:";

//cin>>

numprogram++;

cout << endl << "The program is installed." << endl << "Number installed program::" << numprogram << endl;

}

void uninstall()

{

cout << endl << "Enter the name of the application which you want to uninstall:";

//cin>>

numprogram--;

cout << endl << "The program is uninstalled." << endl << "Number installed program::" << numprogram << endl;

}

void timeneed()

{

int numop;

cout << endl << "Enter the number of operations the program must perform::";

cin >> numop;

cout << endl << "Time for execution(in seconds):" << numop / operation\_per\_seconds << endl;

}

void Showdata()

{

cout << endl << "Cpu model:" << cpumodel;

cout << endl << "Video model:" << videomodel;

cout << endl << "Operation per seconds" <<operation\_per\_seconds;

cout << endl << "Hard size" << hardsize;

cout << endl << "Ram size" << ramsize;

cout << endl << "Time of operation" << optime;

cout << endl << "Number of program" << numprogram;

}

};

int main()

{

string fcpuname = "i7-8400", fvname = "GTX 1660";

CComputer firstpc(fcpuname, fvname, 4000, 3000, 16, 1, 164);

firstpc.Get\_optime();

firstpc.Set\_operation\_per\_seconds(6000);

firstpc.Showdata();

firstpc.timeneed();

firstpc.install();

firstpc.Showdata();

string scpuname = "i7-8400K", svname = "GTX 1060 Ti";

CComputer secondpc(scpuname, svname, 3000, 2000, 12, 2, 104);

secondpc.Get\_cpumodel();

secondpc.Get\_numprogram();

secondpc.Get\_ramsize();

secondpc.Showdata();

secondpc.Set\_hardsize(8000);

secondpc.install();

secondpc.Showdata();

return 0;

}

*Вікно результату:*



*Висновок:* я познайомився з класами, методами, контруктора да деконструкторами.