Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**Звіт з лабораторної роботи №6**

на тему:

“Спадкування ”

**Виконав:**

Ст. гр. КІ-15

Стецик О.І.

**Перевірив:**

Викладач

Козак Н.Б.

Львів – 2020

**Мета роботи:** познайомитися із перевантаженням операторів.

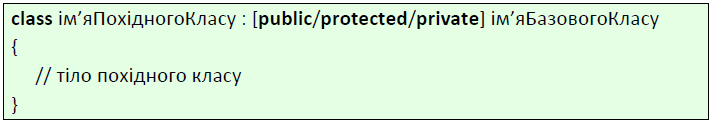
**Теоретичні відомості:**

**Спадкування (ієрархія "іs a")**

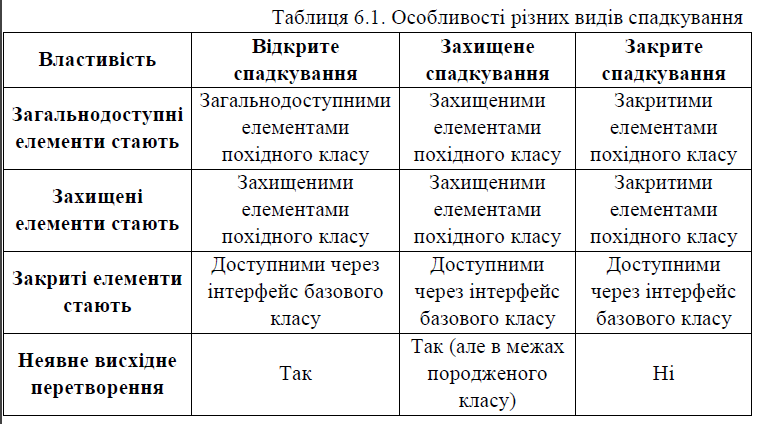
***Спадкування*** – це механізм, за допомогою якого один клас може одержувати атрибути та функціональність іншого. Спадкування дозволяє створювати ієрархію класів.

При створенні нового класу, що повністю дублює існуючий клас і дещо розширяє його новими властивостями і функціональністю програміст може не дублювати існуючий клас і дописувати в нього розширену функціональність, а вказати, що новий клас є спадкоємцем елементів попередньо визначеного класу і визначити у ньому лише необхідну нову функціональність. В цьому випадку існуючий клас, функціональність якого розширюється у новому класі, називається ***базовим класом*** (base class). Новостворений клас називається ***похідним класом*** (derived class), або ***спадкоємцем***. Кожен похідний клас може бути використаним у ролі базового класу для майбутніх похідних класів створюючи при цьому ***дерево спадкування***, яке ще називають ***ієрархією спадкування класів*** (class hierarchy). Спадкування прийнято відображати у вигляді графу (дерева) у напрямку зверху-вниз. При цьому клас, що є у самому верху є самим першим базовим класом і називається ***кореневим класом*** або ***коренем дерева спадкування класів***. Похідний клас, через проміжний, може наслідувати характеристики базового класу. У цьому випадку говорять, що ***базовий клас є непрямим базовим класом (indirect base class) для похідного***. Зокрема, корінь дерева наслідувань є непрямим базовим класом для усіх класів, які знаходяться нижче першого рівня ієрархії. Клас, який При ***одиночному спадкуванні*** (single inheritance) клас породжується одним базовим класом. При ***множинному спадкуванні*** (multiple inheritance) похідний клас успадковує властивості декількох базових класів, причому можлива ситуація коли один базовий клас буде успадкований кілька разів по кількох гілках. При створенні об'єкта похідного класу в пам'яті зберігаються копії усіх класів, які ***становлять вітку, що породила даний клас***.

Похідний клас наслідує характеристики базового через ***специфікатор доступу (acces specifier) : "довкрапка"****.* Нижче наведено синтаксис спадкування базового класу:



За допомогою специфікатора доступу можна визначити, яким чином елементи базового класу будуть успадковуватися похідним класом. При відкритому спадкуванні (використанні специфікатора publіc) у похідному класі члени базового класу мають ті ж специфікатори доступу, що й у базовому класі. При захищеному спадкуванні (використанні специфікатора protected) у похідному класі відкриті члени базового класу стають захищеними, а інші зберігають своє початкове значення специфікатора доступу. Нарешті, при закритому спадкуванні (використанні специфікатора prіvate) у похідному класі всі члени базового класу стають закритими.



***Завдання:***

Створити абстрактний базовий клас і похійдний від нього клас, які реалізують модель предметної області згідно варіанту. Кожен клас має мати мінімум 3 власні елементи даних один з яких створюється динамічно, методи встановлення і читання характеристик елементів-даних класу (Set і Get), та мінімум 2 абстрактні методи обробки даних і мінімум 2 методи обробки даних у похідному класі. Крім цього клас має містити перевантаження оператора присвоєння, конструкторів по замовчуванню і копіювання та віртуальний деструктор. Для розроблених класів реалізувати програму-драйвер, яка демонструє роботу класів.

******

*Код програми:*

#include <iostream>

#include <conio.h>

#include "Header.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

Boat victory;

GuardBoat police;

victory.set\_weight(30);

victory.show\_weight();

victory.set\_speed(15);

victory.show\_speed();

police.set\_height(12);

police.show\_height();

police.set\_weight(50);

police.show\_weight();

system("pause");

return 0;

}

**Header:**

#pragma once

#include <string.h>

#include <iostream>

using namespace std;

class Boat {

private:

int weight;

int speed;

public:

Boat(int Weight, int Speed)

{

this->weight = Weight;

this->speed = Speed;

};

Boat()

{

this->weight = 1;

this->speed = 10;

};

Boat(const Boat& copy) :

weight(copy.weight), speed(copy.speed)

{

cout << "Copy constructor worked here...";

}

Boat& operator= (const Boat& drob)

{

weight = drob.weight;

speed = drob.speed;

return \*this;

}

void set\_weight(int Weight) {

this->weight = Weight;

}

int get\_weight() {

return this->weight;

}

void set\_speed(int Speed) {

this->speed = Speed;

}

char get\_speed() {

return this->speed;

}

void show\_weight() {

cout << "Weight is: " << this->weight <<" kg"<< endl;

}

void show\_speed() {

cout << "Speed is " << this->speed <<" km/h"<< endl;

}

~Boat()

{

};

};

class GuardBoat : public Boat {

private:

double height;

double capacity;

public:

GuardBoat()

{

this->height = 100;

this->capacity = 7;

};

GuardBoat(int Height, int Capacity)

{

this->height = Height;

this->capacity = Capacity;

};

void set\_height(double Height) {

this->height = Height;

}

double get\_height()

{

return this->height;

}

void show\_height() {

cout << "A height of Guard Boat is: " << this->height << " m" << endl;

}

void set\_capacity(double Capacity) {

this->capacity = Capacity;

}

double get\_capacity()

{

return this->capacity;

}

void show\_capacity() {

cout << "A capacityof Guard Boat is: " << this->capacity << " humans" << endl;

}

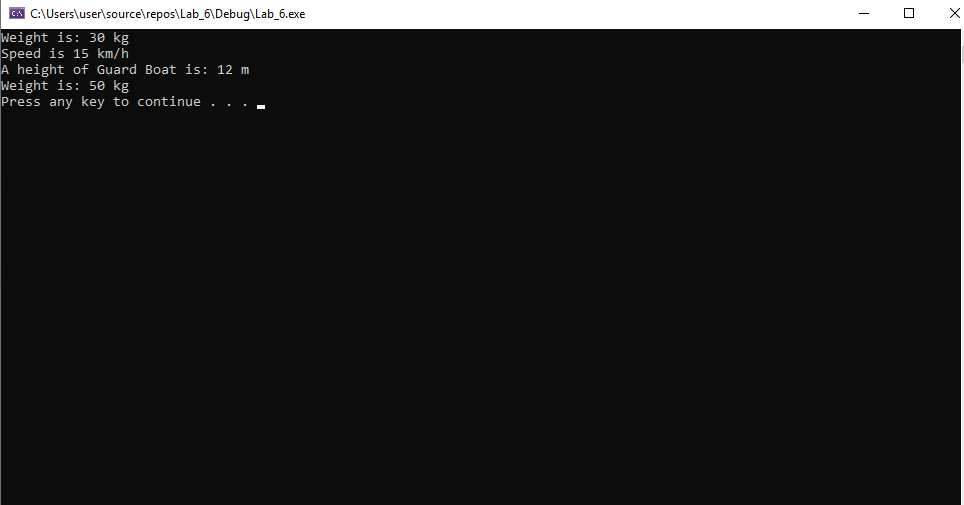
virtual ~GuardBoat()

{

};

};

*Вікно результату:*



*Висновок:* я познайомився з спадкуванням класів, з абстрактними класами, та віртуальними методами класу.