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

Зектер В.О.

**Перевірив:**

Викладач

Козак Н.Б.

Львів 2020

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

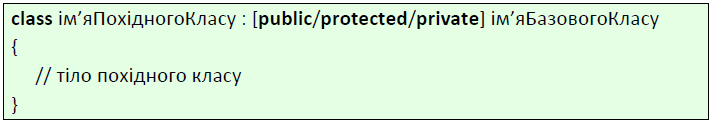
**Теоретичні відомості:**

**Спадкування (ієрархія "іs a")**

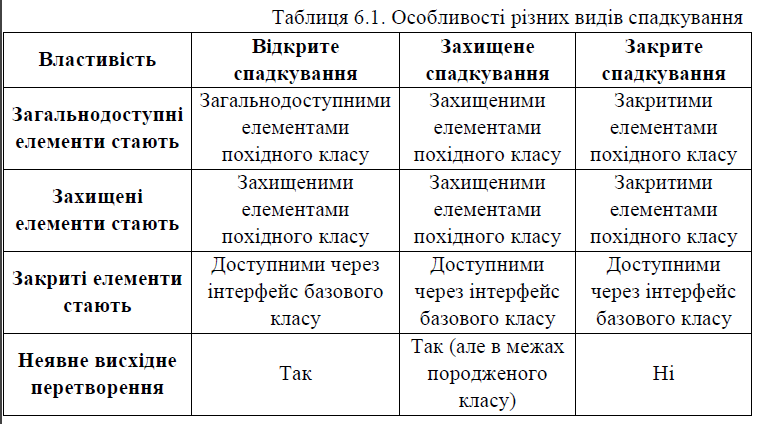
***Спадкування*** – це механізм, за допомогою якого один клас може одержувати атрибути та функціональність іншого. Спадкування дозволяє створювати ієрархію класів.

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:

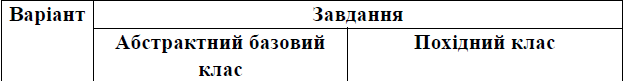


За допомогою специфікатора доступу можна визначити, яким чином елементи базового класу будуть успадковуватися похідним класом. При відкритому спадкуванні (використанні специфікатора publіc) у похідному класі члени базового класу мають ті ж специфікатори доступу, що й у базовому класі. При захищеному 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>

using namespace std;

class Pristriy {

private:

int vaga;

int vartist;

public:

Pristriy(int Vaga, int Vartist)

{

this->vaga = Vaga;

this->vartist = Vartist;

};

Pristriy()

{

this->vaga = 3;

this->vartist = 2700;

};

Pristriy(const Pristriy& copy) :

vaga(copy.vaga), vartist(copy.vartist)

{

cout << "Вызвалсся конструктор копирования";

}

Pristriy& operator= (const Pristriy& drob)

{

vaga = drob.vaga;

vartist = drob.vartist;

return \*this;

}

void set\_vaga(int Vaga) {

this->vaga = Vaga;

}

int get\_vaga() {

return this->vaga;

}

void set\_vartist(int Vartist) {

this->vartist = Vartist;

}

char get\_vartist() {

return this->vartist;

}

void show\_vaga() {

cout << "Вес: " << this->vaga << " килограм" << endl;

}

void show\_vartist() {

cout << "Стоимость " << this->vartist << endl;

}

~Pristriy()

{

};

};

class Monitor : public Pristriy {

private:

double visota;

double kilkist\_duymiv;

public:

Monitor()

{

this->visota = 300;

this->kilkist\_duymiv = 26;

};

Monitor(int Visota, int Kilkist\_duymiv)

{

this->visota = Visota;

this->kilkist\_duymiv = Kilkist\_duymiv;

};

void set\_visota(double Visota) {

this->visota = Visota;

}

double get\_visota()

{

return this->visota;

}

void show\_visota() {

cout << "Высота монитора " << this->visota << " см" << endl;

}

void set\_kilkist\_duymiv(double Kilkist\_duymiv) {

this->kilkist\_duymiv = Kilkist\_duymiv;

}

double get\_kilkist\_duymiv()

{

return this->kilkist\_duymiv;

}

void show\_kilkist\_duymiv() {

cout << "Количество дюймов: " << this->kilkist\_duymiv << endl;

}

virtual ~Monitor()

{

};

};

int main() {

setlocale(LC\_ALL, "Russian");

Pristriy monik;

Monitor samsung;

monik.set\_vaga(2);

monik.show\_vaga();

monik.set\_vartist(4500);

monik.show\_vartist();

samsung.set\_visota(360);

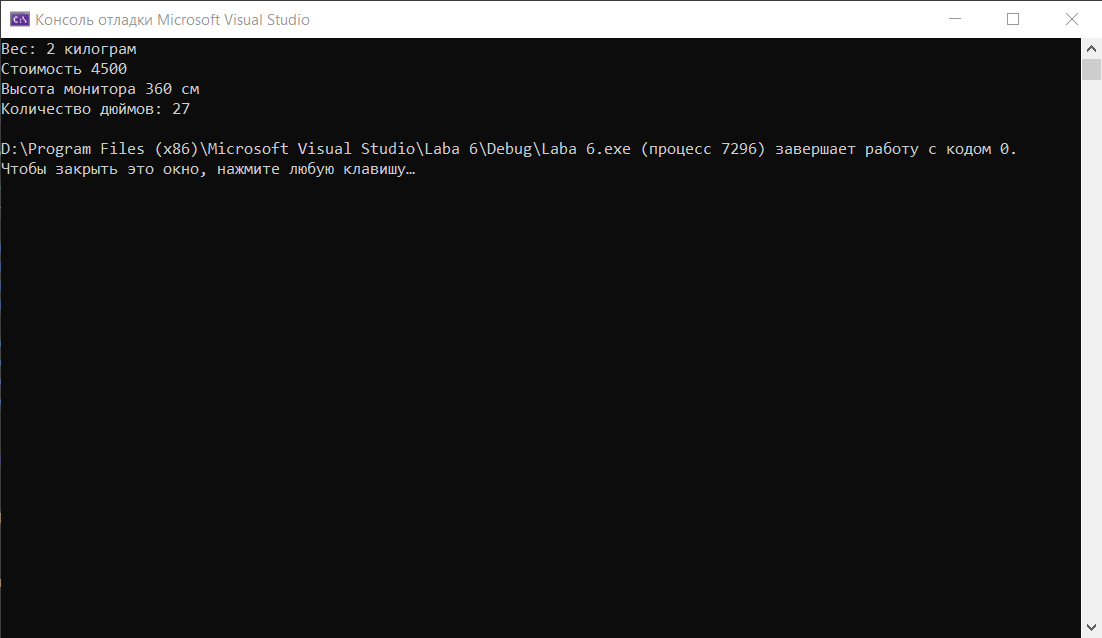
samsung.show\_visota();

samsung.set\_kilkist\_duymiv(27);

samsung.show\_kilkist\_duymiv();

return 0;

}

**Вікно результату: **

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