Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**Звіт з лабораторної роботи №6**

**Програмування, частина 2**

**(Об’єктно - орієнтоване програмування)**

на тему:

“Спадкування”

Варіант - 25

**Виконала:**

ст. гр. КІ-15

Романішина А. Ю.

**Перевірив:**

асист. каф. ЕОМ

Козак Н. Б.

Львів – 2020

**Мета:** познайомитися із спадкуванням класів.

**Теоретичні відомості:**

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.

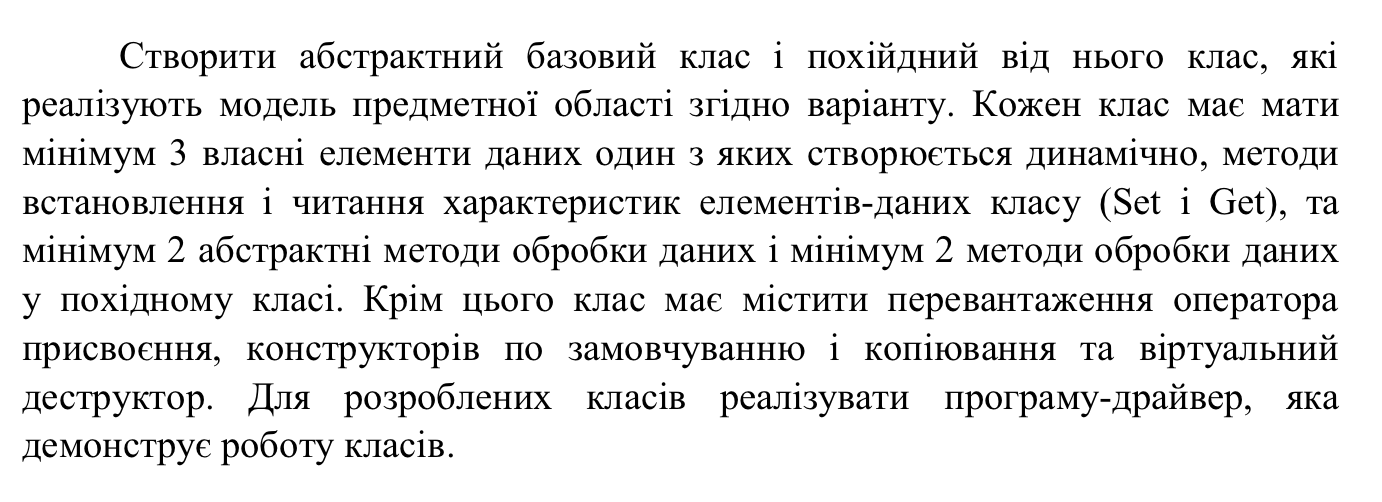
При створенні об'єкту класу автоматично викликається його конструктор. Якщо об'єкт породжується від похідного класу, то, очевидно, при створенні об‘єкту похідного класу повинен викликатись і конструктор базового класу. У цьому випадку порядок виклику і виконання конструкторів є таким:

1. при створенні об'єкта похідного класу генерується виклик конструктора цього класу, але сам конструктор не виконується;
2. після генерації виклику конструктора похідного класу генерується виклик і виконується конструктор базового класу;
3. по закінченню виконання тіла конструктора базового класу виконується тіло конструктора похідного класу.

У випадку багатократного спадкування наведений порядок виклику і виконання конструкторів зберігається. Глибина вкладень конструкторів при цьому може бути довільною. Порядок виклику конструкторів є послідовним по мірі ієрархічного спадкування і строго фіксується.

За допомогою специфікатора доступу у визначенні конструктора похідного класу можна передати параметри конструктору базового класу. Очевидно, при цьому базовий клас повинен містити відповідний конструктор, який здатний прийняти такий параметр.

**Індивідуальне завдання:**

****

**Код програми:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <string.h>

#include <iostream>

#include <stdlib.h>

#include <time.h>

#include <iostream>

#include <Windows.h>

**using** **namespace** std;

**class** Conditioner

{

**public**:

Conditioner(**const** **char**\* set\_Model)

{

Model = **new** **char**[strlen(set\_Model) + 1];

strcpy(Model, set\_Model);

}

Conditioner()

{

Model = **new** **char**[strlen("None") + 1];

strcpy(Model, "None");

}

**virtual** ~Conditioner()

{

**delete**[] Model;

}

**virtual** **void** setModel(**const** **char**\* set\_Model)

{

Model = **new** **char**[strlen(set\_Model) + 1];

strcpy(Model, set\_Model);

}

**virtual** **void** ON()

{

IsWork = 1;

}

**virtual** **void** OFF()

{

IsWork = 0;

}

**virtual** **char**\* Mode() = 0;

**virtual** **void** setMode(**bool** k) = 0;

**virtual** **bool** Status()

{

**return** IsWork;

}

**protected**:

**char**\* Model;

**bool** IsWork = 0;

};

**class** ClimateControl : **public** Conditioner

{

**public**:

ClimateControl(**const** **char**\* set\_Model) : Conditioner(set\_Model)

{

}

ClimateControl() : Conditioner()

{

}

**virtual** ~ClimateControl()

{

}

**virtual** **void** Set\_Temperature(**int** Temperature)

{

**this**->Temperature = Temperature;

}

**virtual** **char**\* Mode()

{

**char**\* curMode = **new** **char**[10];

**if** (mode == 0)

{

strcpy(curMode, "Colling");

}

**else**

{

strcpy(curMode, "Heating");

}

**return** curMode;

}

**virtual** **void** setMode(**bool** k)

{

mode = k;

}

**virtual** **int** Get\_Temperature()

{

**return** Temperature;

}

ClimateControl& **operator** = (**const** ClimateControl& Climate)

{

**this**->Temperature = Climate.Temperature;

**this**->mode = Climate.mode;

**this**->Model = Climate.Model;

**this**->IsWork = Climate.IsWork;

**return** \***this**;

}

**private**:

**int** Temperature = 0;

**bool** mode = 0;

};

**int** main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

ClimateControl Control("Samsung"), Control2;

Control.Set\_Temperature(25);

Control.setMode(1);

Control2 = Control;

Control.Set\_Temperature(20);

Control.setMode(0);

cout << "Control info:" << endl;

cout << "Поточна температура: " << Control.Get\_Temperature() << endl;

cout << "Поточний режим: " << Control.Mode() << endl;

cout << "Control2 info:" << endl;

cout << "Поточна температура: " << Control2.Get\_Temperature() << endl;

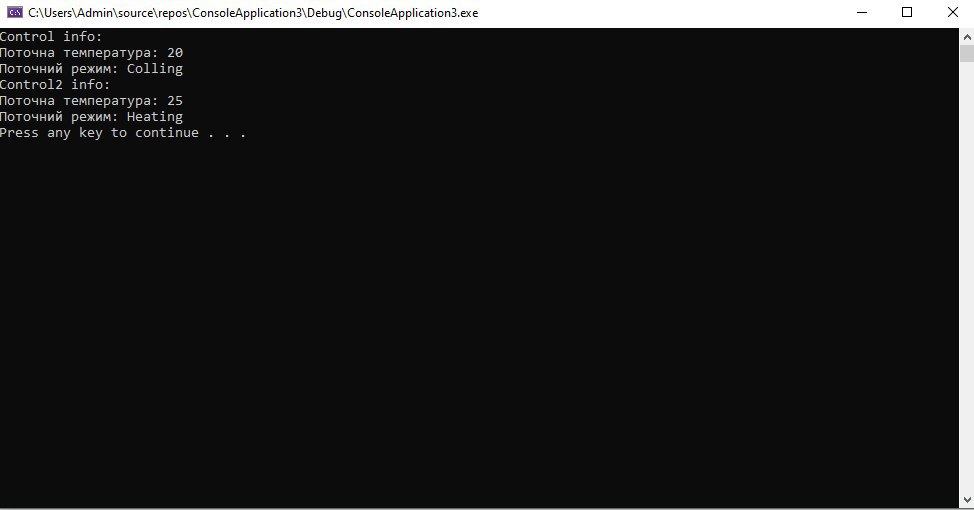
cout << "Поточний режим: " << Control2.Mode() << endl;

system("pause");

**return** 0;

}

**Результат виконання функції :**

****

**Висновок:** під час виконання цієї лабораторної роботи я навчилась працювати зі спадкуванням класів.