Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №6

з дисципліни: “Програмування, ч.2 (ООП)”

на тему: “Cпадкування”

Варіант 19

Виконав: ст.гр. КІ-15

Мармура В.І.

Прийняла: Викладач

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

Завдання (Варіант 19) -

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.

Машина - Бронетранспортер

Код для вирішення задачі даного варіанту -

#include <iostream>

using namespace std;

class Car {

public:

int getSpeed() {

return Speed;

}

int getmaxPax() {

return maxPax;

}

int getmaxSpeed() {

return maxSpeed;

}

void setSpeed(int speed) {

Speed = speed;

}

void setmaxPax(int pax) {

maxPax = pax;

}

void setmaxSpeed(int speed) {

maxSpeed = speed;

}

virtual void Move () = 0;

virtual void offPass() = 0;

virtual ~Car() {

cout << "Deleted" << endl;

}

protected:

int Speed, maxPax, maxSpeed;

};

class Armored\_Car :public Car {

public:

int getArmor() {

return Armor;

}

bool getisSwim() {

return isSwim;

}

bool getisWeapon() {

return isWeapon;

}

void setArmor(int pow) {

Armor = pow;

}

void setisSwim(bool is) {

isSwim = is;

}

void setisWeapon(bool is) {

isWeapon = is;

}

void Shooted(int damage) {

if (Armor < damage) {

cout << "Vehicle is destroyed." << endl;

}

else Armor -= damage;

}

virtual void Move() override{

cout << "Armored Car is moving." << endl;

}

virtual void offPass() override {

cout << "Armored Car has its pax off." << endl;

}

Armored\_Car operator= (const Armored\_Car &armcar) {

this->Armor = armcar.Armor;

this->isSwim = armcar.isSwim;

this->isWeapon = armcar.isWeapon;

this->maxPax = armcar.maxPax;

this->maxSpeed = armcar.maxSpeed;

this->Speed = 0;

}

void Swim(int dist) {

//swim

if (isSwim == true) {

cout << "Vehicle is swimming." << endl;

}

else cout << "Vehicle cannot swim and destroyed." << endl;

}

Armored\_Car() {

Armor = 0;

isSwim = false;

isWeapon = false;

Speed = 0;

maxPax = 2;

maxSpeed = 1;

}

Armored\_Car(const Armored\_Car &armcar) {

this->Armor = armcar.Armor;

this->isSwim = armcar.isSwim;

this->isWeapon = armcar.isWeapon;

this->maxPax = armcar.maxPax;

this->maxSpeed = armcar.maxSpeed;

this->Speed = 0;

}

Armored\_Car(int speed, int armor, int max\_pax, bool swim, bool weap ) {

Speed = speed;

Armor = armor;

maxPax = max\_pax;

isSwim = swim;

isWeapon = weap;

}

private:

int Armor;

bool isSwim, isWeapon;

};

int main()

{

cout << "Enter a speed of vehicle - " << endl;

int speed;

cin >> speed;

cout << "Enter an armour of vehicle - " << endl;

int arm;

cin >> arm;

Armored\_Car armcar(speed, arm, 15, true, true);

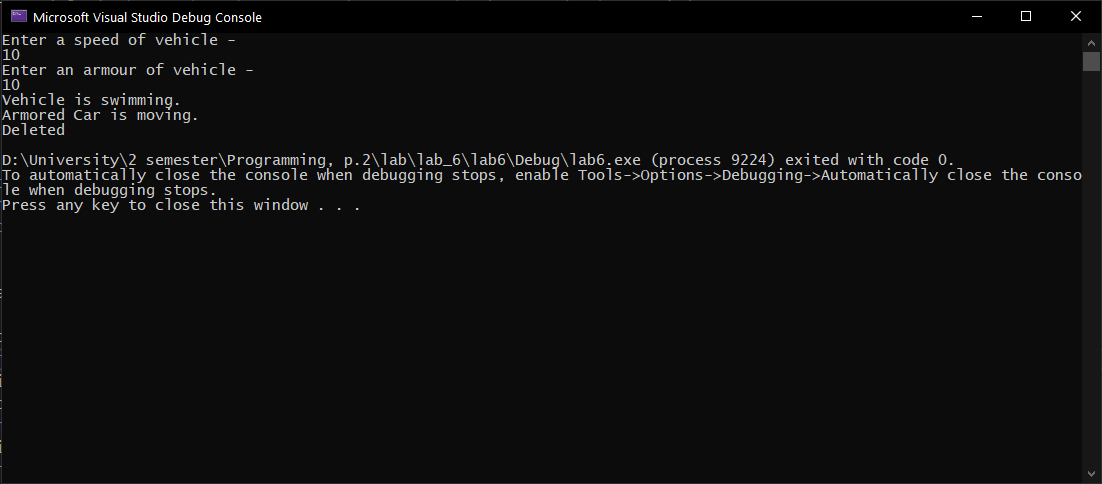
armcar.Swim(1);

armcar.Move();

return 0;

}

Скріншот виконання програми –



Висновок : в даній лабораторній роботі я опрацював роботу з спадкуванням, абстрактними классами, застосуванням їх і використанням віртуальних функцій.