 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №3**

з дисципліни «Об’єктно-орієнтоване програмування»

на тему: «Успадкування. Поліморфізм»

Варіант №7

**Виконав:**

студент гр. БС-82

Грініх К. А.

**Перевірив:**

ас. каф. БМК

Рисін С. В.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2020

**🞏 Практичне заняття без зауважень**

**🞏 Практичне заняття має зауваження:**

**🞏 несвоєчасний захист**

**🞏 присутні зауваження до UML діаграми:**

**🞏 діаграма класу не відповідає коду**

**🞏 виконані не за стандартом:**

**🞏 атрибути**

**🞏 відношення**

**🞏 потужність**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 присутні зауваження до коду:**

**🞏 задача завдання вирішена хибно**

**🞏 код програми не компілюється**

**🞏 хибно задані специфікатори доступу**

**🞏 помилки у визначенні конструкторів / деструкторів**

**🞏 відсутні списки ініціалізації в конструкторах**

**🞏 константні методи**

**🞏 використано глобальні змінні**

**🞏 статичні змінні при роботі з масивами**

**🞏 оформлення коду**

**🞏 присутні зайві символи «{» та «}»**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 невірні відповіді на запитання:**

**🞏 №1 🞏 №2 🞏 №3 🞏 №4 🞏 №5**

**🞏 №6 🞏 №7 🞏 №8 🞏 №9 🞏 №10**

**🞏 маються інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Завдання:**

1. Вивчити теоретичні основи визначення класів, що використовують спадкування властивостей, та побудови ієрархії класів в мові С++.
2. Розробити діаграму класів в нотації UML та програмний застосунок з використанням успадкування, що реалізує ієрархію класів відповідно до свого варіанту (визначення класів та їх реалізації мають бути розташовані у файлах \*.h та \*.cpp відповідно; для ініціалізації даних класів використовувати введення з клавіатури):

Базовий клас: засіб пересування (назва, швидкість, компанія виробник)

Класи нащадки: Вантажний автомобіль (вантажопідйомність), Автобус (кількість пасажирів)

Завдання: використовуюи віртуальні функції, не знаючи з об’єктом якого класу ви працюєте, виведіть на екран швидкість та компанію виробника для засобу пересування.

1. Скласти і захистити звіт по роботі.

**Діаграма:**



**Лістинг програми:**

Файл Vehicle.h:

#pragma once

#include <string>

#include <iostream>

using namespace std;

class Vehicle

{

protected:

string name;

unsigned short speed;

string company;

public:

Vehicle(string Name, unsigned short Speed, string Company);

virtual ~Vehicle();

string get\_name() const;

unsigned short get\_speed() const;

string get\_company() const;

void set\_name(string Name);

void set\_speed(unsigned short Speed);

void set\_company(string Company);

virtual string get\_obj\_name() const;

virtual string get\_obj\_info() const;

void Screen() const;

};

Файл Vehicle.cpp:

#include <iostream>

#include <string>

using namespace std;

#include "Vehicle.h"

Vehicle::Vehicle(string Name, unsigned short Speed, string Company) : name(Name), speed(Speed), company(Company) {}

Vehicle::~Vehicle(){}

string Vehicle::get\_name() const { return name; }

unsigned short Vehicle::get\_speed() const { return speed; }

string Vehicle::get\_company() const { return company; }

void Vehicle::set\_name(string Name) { name = Name; }

void Vehicle::set\_speed(unsigned short Speed) { speed = Speed; }

void Vehicle::set\_company(string Company) { company = Company; }

string Vehicle::get\_obj\_name() const

{

return ("Vehicle").c\_str();

}

string Vehicle::get\_obj\_info() const

{

return "";

}

void Vehicle::Screen() const

{

cout <<"Name of object: "<< get\_obj\_name() <<endl;

cout << "The speed of the vehicle: " << get\_speed() << endl;

cout << "The producer company of the vehicle: " << get\_company() << endl;

cout << get\_obj\_info() << endl;

}

Файл Truck.h:

#include <string>

using namespace std;

#include "Vehicle.h"

class Truck : public Vehicle

{

private:

unsigned short capacity;

public:

Truck(string Name, unsigned short Speed, string Company, unsigned short Capacity);

~Truck();

int get\_capacity() const;

void set\_capacity(unsigned short Capacity);

string get\_obj\_name() const;

string get\_obj\_info() const;

};

Файл Truck.cpp:

#include <iostream>

#include <string>

#include "Truck.h"

Truck::Truck(string Name, unsigned short Speed, string Company, unsigned short Capacity):

Vehicle(Name, Speed, Company), capacity(Capacity) {}

Truck::~Truck() {}

int Truck::get\_capacity() const { return capacity; }

void Truck::set\_capacity(int Capacity) { capacity = Capacity; }

string Truck::get\_obj\_name() const

{

return ("Truck").c\_str();

}

string Truck::get\_obj\_info() const

{

return "The capacity of vehicle"+to\_string(get\_capacity());

}

Файл Bus.h:

#include "Vehicle.h"

class Bus : public Vehicle

{

private:

unsigned short passengers;

public:

Bus(string Name, unsigned short Speed, string Company, unsigned short Passengers);

~Bus();

unsigned short get\_passengers() const;

void set\_passengers(unsigned short Passengers);

string get\_obj\_name() const;

string get\_obj\_info() const;

};

Файл Bus.cpp:

#include <iostream>

#include <string>

#include "Bus.h"

Bus::Bus(string Name, unsigned short Speed, string Company, unsigned short Passengers):

Vehicle(Name, Speed, Company), passengers(Passengers) {}

Bus::~Bus() {}

int Bus::get\_passengers()const { return passengers; }

void Bus::set\_passengers(unsigned short Passengers) { passengers = Passengers; }

string Bus::get\_obj\_name() const

{

return "Bus";

}

string Bus::get\_obj\_info() const

{

return "The capacity of vehicle"+to\_string(get\_capacity());

}

Файл main.cpp:

#include "Vehicle.h"

#include "Truck.h"

#include "Bus.h"

#include <iostream>

#include <string>

using namespace std;

void main()

{

const int elems = 3;

Vehicle \* array[elems];

for (int i = 0; i < elems; i++)

{

string name, company;

unsigned short speed=0;

cout << "---------------------------------" << endl;

cout << "Print the name of velicle: ";

cin >> name;

cout << "Print the speed of velicle: ";

cin >> speed;

cout << "Print the producer company of velicle: ";

cin >> company;

switch (i)

{

case 0:

{

array[0] = new Vehicle(name, speed, company);

}

break;

case 1:

{

unsigned short capacity=0;

cout << "Print the capacity: ";

cin >> capacity;

array[1] = new Truck(name, speed, company, capacity);

}

break;

case 2:

{

unsigned short passengers=0;

cout << "Print the number of passengers: ";

cin >> passengers;

array[2] = new Bus(name, speed, company, passengers);

}

break;

default:

{

cout << "Error!" << endl;

}

continue;

}

}

for (int i = 0; i < elems; i++)

{

cout << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*The result of the program: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

array[i]->Screen();

}

for (int i = 0; i < elems; ++i)

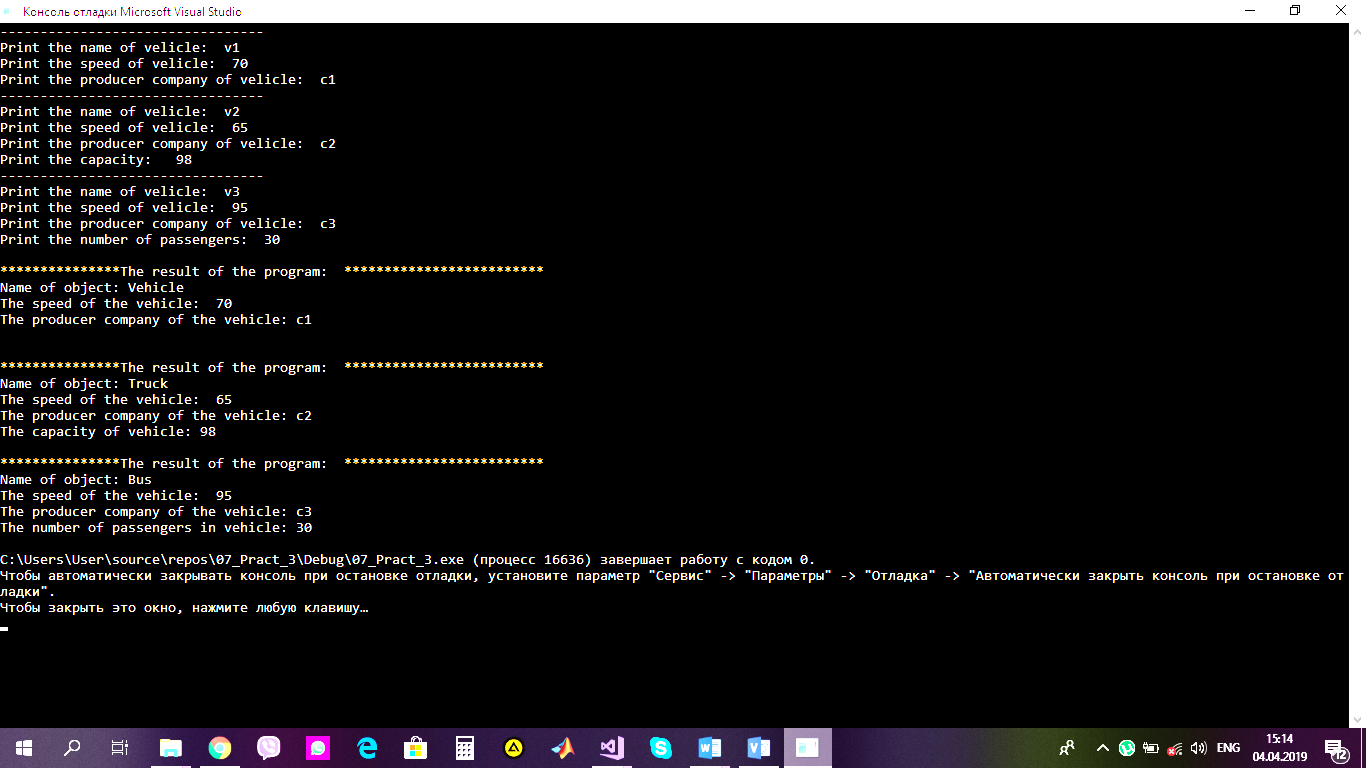
{

delete array[i];

}

}

**Скріншот результатів виконання:**



**Відповіді на контрольні запитання:**

1. Що означає поняття спадкування?

Успадкування – механізм створення нових класів, які називаються нащадками або похідними класами і створюються із вже існуючих так званих базових класів. При цьому клас-нащадок набуває властивостей базового класу.

1. Який клас називається базовим, а який клас нащадком?

У програмуванні базовим класом називають клас, що знаходиться нижче в ієрархії дерева класів, ніж його нащадок. Класом-нащадком називається класс, який успадковує поля і методи базового класу.

1. Скільки базових класів може мати похідний клас?

1 або більше.

1. Чи може похідний клас бути базовим?

Так, він буде базовим для свого класу-нащадка, який буде знаходитись нижче в ієрархії класів.

1. Як змінюється доступ до елементів базового класу при спадкуванні з різними специфікаторами доступу: з розділів класу, із програми, з інших класів?

public-специфікатор спадкування:

Всі public і protected члени базового класу доступні і залишаються public і protected членами похідного класу.

protected-специфікатор спадкування:

Всі public i protected члени базового класу доступні і стають protected-членами похідного класу, private-секція не має доступу.

private-специфікатор спадкування:

protected i public доступні і стають private членами похідного класу.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | базовий | | | |
|  | public | protected | private |
| нащадок | public | + public | +  protected | - |
| protected | + protected | + protected | - |
| private | + private | + private | - |

Незалежно від специфікатору доступу при спадкуванні, доступ із програми та з інших класів можливий тільки до елементів із public секції. Проте, доступ до елементів з усіх секцій може бути реалізований за допомогою дружніх функцій, які оголошуються з ключовим словом **friend.**

1. У чому різниця між простим і множинним спадкуванням?

Просте наслідування описує спорідненість між двома класами: один з яких наслідує другий. Клас, що знаходиться на вершині ієрархії, називається базовим класом. З одного класу може виводитись багато класів, але навіть в цьому випадку подібний спадкування залишається простим.

Множинне успадкування – побудова похідного класу на основі декількох базових класів. В такому випадку замість імені одного базового класу використовується список імен через кому, наприклад:

class A { /\*…\*/ };

class B { /\*…\*/ };

class C: public A, private B { /\*…\*/ };

1. Чи можна з класу-нащадка одержати доступ до private частин базового класу, якщо специфікатор доступу при спадкуванні private?

Ні, це неможливо.

1. Чи успадковуються конструктори, деструктори?

Так, успадковуються. До того ж, можна викликати конструктор та деструктор базового класу при декларації конструктору класу-нащадка.

1. Що таке віртуальна функція?

Віртуальна функція – метод класу, який може бути перевизначений в класах-нащадках так, що конкретна реалізація методу для виклику буде визначатися під час виконання. Таким чином, не обов’язково знати тип об’єкта для роботи з ним через віртуальні методи, так як викликається метод не із класу типу самого вказівника, а із класу типу об’єкта.

1. Чи можна віртуальну функцію визначити як static, відповідь пояснити.

Не можна, через те, що специфікатор static означає приналежність до класу, не до його конкретної сутності.