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 №5

**Виконав:**

студент гр. БС-71

Воробйов І.О.

**Перевірив:**

асистент каф. БМК

Рисін С.В.

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

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

(підпис викладача)

Київ-2019

**🞏 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:**

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

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

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

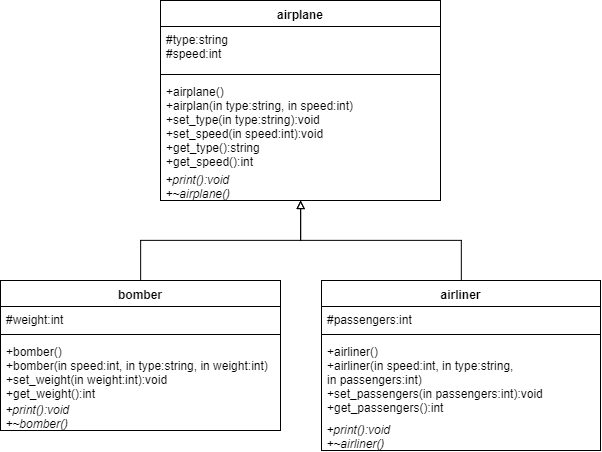
**Завдання:**

1. Вивчити теоретичні основи визначення класів, що використовують спадкування властивостей, та побудуви ієрархії класій в мові С++.
2. Розробити діаграму класів в нотації UML та програмний додаток 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* | *Класи нащадки* | *Завдання* |
| Літак (тип, швидкість) | Пасажирський літак (кількість пасажирів) | Використовуючи віртуальні функції, не знаючи з об’єктом якого класу ви працюєте, виведіть на екран тип і швидкість літака. |
| Бомбардувальник (тоннаж бомбового відсіку) |

1. Скласти і захистити звіт.

**UML діаграма**



**Код програми:**

**airplane.h:**

#pragma once

#include <iostream>

#include <string>

using namespace std;

class airplane

{

protected:

string type;

int speed;

public:

airplane();

airplane(string, int);

void set\_type(string);

void set\_speed(int);

string get\_type() const;

int get\_speed() const;

virtual void print() const;

virtual ~airplane();

};

**airplane.cpp:**

#include "airplane.h"

airplane::airplane() {};

airplane::airplane(string str, int tmp) :type(str), speed(tmp) {}

airplane::~airplane() {};

void airplane::print() const

{

cout << "type = " << type << endl;

cout << "speed = " << speed << endl;

}

void airplane::set\_type(string str)

{

type = str;

}

void airplane::set\_speed(int sp)

{

speed = sp;

}

string airplane::get\_type() const

{

return type;

}

int airplane::get\_speed() const

{

return speed;

}

**bomber.h:**

#pragma once

#include "airplane.h"

class bomber :public airplane

{

private:

int weight;

public:

bomber();

bomber(int, string, int);

void set\_weight(int);

int get\_weight() const;

virtual void print() const;

virtual ~bomber();

};

**bomber.cpp:**

#include "bomber.h"

bomber::bomber(): weight(0)

{

type = "type";

speed = 0;

}

bomber::bomber(int sp, string str, int wg) : weight(wg)

{

speed = sp;

type = str;

}

void bomber::set\_weight(int tmp)

{

weight = tmp;

}

int bomber::get\_weight() const

{

return weight;

}

void bomber::print() const

{

airplane::print();

cout << "weight = " << weight << endl;

}

bomber::~bomber() {}

**airliner.h:**

#pragma once

#include"airplane.h"

class airliner :public airplane

{

private:

int passengers;

public:

airliner();

airliner(int, string, int);

void set\_passengers(int);

int get\_passengers() const;

virtual void print() const;

virtual ~airliner();

};

**airliner.cpp:**

#include "airliner.h"

airliner::airliner() : passengers(0)

{

type = "type";

speed = 0;

}

airliner::airliner(int sp, string str, int ps) : passengers(ps)

{

speed = sp;

type = str;

}

void airliner::set\_passengers(int tmp)

{

passengers = tmp;

}

int airliner::get\_passengers() const

{

return passengers;

}

void airliner::print() const

{

airplane::print();

cout << "passengers = " << passengers << endl;

}

airliner::~airliner() {}

**Source.cpp:**

#include "airliner.h"

#include "bomber.h"

int main()

{

int size\_airliner, size\_bomber;

cout << "Input numb of airliners = ";

cin >> size\_airliner;

cout << "Input numb of bombers = ";

cin >> size\_bomber;

int size = size\_airliner + size\_bomber;

airplane \*\*obj = new airplane\*[size];

for (size\_t i = 0; i < size; i++)

{

string type;

int speed;

cout << "Type = "; cin >> type;

cout << "Speed = "; cin >> speed;

if (i<size\_airliner)

{

int passengers;

cout << "Passengers = "; cin >> passengers;

obj[i] = new airliner(speed, type, passengers);

}

else

{

int weight;

cout << "weight = "; cin >> weight;

obj[i] = new bomber(speed, type, weight);

}

}

for (size\_t i = 0; i < size; i++)

{

obj[i]->print();

cout << endl;

}

for (size\_t i = 0; i < size; i++)

delete obj[i];

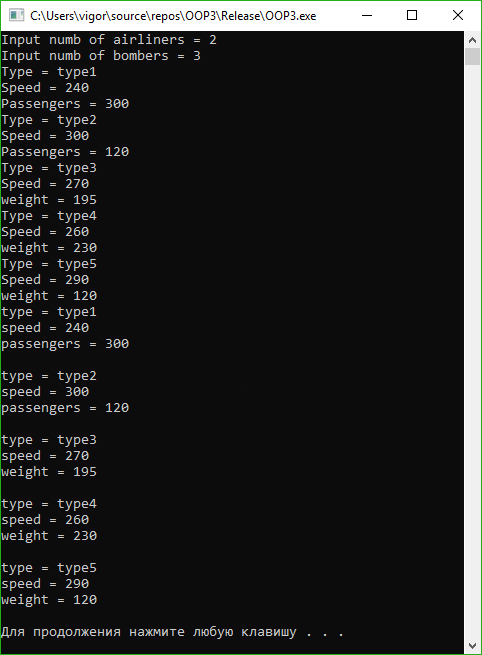
delete[]obj;

system("pause");

return 0;

}

**Результат роботи програми:**



**Контрольні запитання**

1. Що означає поняття спадкування?

Наследование- это свойство с помощью которого один объект может использовать свойства другого.

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

Базовый класс – это класс у которого класс потомок перенимает все поля и свойства, а потомок тот ктоо перенимает.

1. Скільки базових класів може мати похідний клас?

Сколько угодно.

1. Чи може похідний клас бути базовим?

Да, но не для тогоже класса.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Модификатор наследования→** | **private** | **protected** | **public** |
| **Модификатор доступа ↓** |
| **public** | **private** | **protected** | **public** |
| **protected** | **private** | **protected** | **protected** |
| **private** | закрыт | закрыт | закрыт |

Доступ из программы и из других классов получают только public при наследовании public.

1. У чому різниця між простим і множинним спадкуванням?

Множественное наследование имеет несколько базовых классов.

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

Нет.

1. Чи успадковуються конструктори, деструктори?

Нет.

1. Що таке віртуальна функція?

Виртуальная функция – это функция которая может быть переопределена в классе потомке и вызоваться будет в зависимости от типа объекта, а не указателя.

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

Не может. Так как у статической функции нет this.