Bài 1

1. c=1/3 + 8=25/3

++c: c==28/3

a= 1/3 + 28/3

a= 29/3

b=29/3

PhanSo();

PhanSo(const int&,const int&);

PhanSo operator+(const Phanso&);

PhanSo operator+=(const Phanso&);

PhanSo operator+(const int&);

PhanSo operator++();

Operator int(int);

Friend ostream& operator <<(ostream&, const PhanSo&);

~PhanSo();

Câu 2

Khái niệm đa hình (polymorphism) là một trong những nguyên tắc quan trọng của lập trình hướng đối tượng (OOP). Đa hình cho phép các đối tượng của các lớp khác nhau có thể hiện thực một phương thức cụ thể một cách khác nhau.

Ví dụ code

#include <iostream>

using namespace std;

class Animal {

public:

    virtual void speak() {

        cout << "This is an animal speaking." << endl;

    }

};

class Dog : public Animal {

public:

    void speak() {

        cout << "Woof! Woof!" << endl;

    }

};

class Cat : public Animal {

public:

    void speak() {

        cout << "Meow! Meow!" << endl;

    }

};

class Duck : public Animal {

public:

    void speak() {

        cout << "Quack! Quack!" << endl;

    }

};

// Sử dụng đa hình

void animal\_sound(Animal\* animal) {

    animal->speak();

}

int main() {

    Dog dog;

    Cat cat;

    Duck duck;

    animal\_sound(&dog);  // In ra "Woof! Woof!"

    animal\_sound(&cat);  // In ra "Meow! Meow!"

    animal\_sound(&duck); // In ra "Quack! Quack!"

    return 0;

}

Lớp trừu tượng (abstract class) trong lập trình hướng đối tượng là một lớp mà không thể được khởi tạo dưới dạng một đối tượng cụ thể. Lớp trừu tượng thường chứa các phương thức trừu tượng(abstract methods)- các phương thức chỉ được khai báo mà không có triển khai cụ thể.

Code ví dụ

#include <iostream>

using namespace std;

// Khai báo lớp trừu tượng

class AbstractClass {

public:

    // Phương thức trừu tượng

    virtual void abstractMethod() = 0;

};

// Lớp con kế thừa từ lớp trừu tượng

class ConcreteClass : public AbstractClass {

public:

    void abstractMethod() {

        cout << "Concrete class implementing abstract method." << endl;

    }

};

int main() {

    // Không thể khởi tạo đối tượng của lớp trừu tượng

    // AbstractClass abstractObj; // Lỗi

    ConcreteClass concreteObj;

    concreteObj.abstractMethod(); // In ra "Concrete class implementing abstract method."

    return 0;

}

Câu 3

Ý nghĩa như sau:

Từ khóa virtual: Giúp cái lớp dẫn xuất có thể thừa kế lại hoàn toàn tính chất này hoặc có thể viết chồng ngay lên

Gán bằng 0: Bắt buộc dẫn xuất phải viết lại tính lại, nếu không viết lại sẽ báo lỗi cú pháp.