Họ và tên : Đỗ Thái Dương  
MSSV : N22DCDK018  
Lớp : D22CQDK01-N

**Đề 2012**

**Câu 1:**

A)

Kết quả xuất ra như sau:

c = 1 / 3 + 8 = 25 / 3

++c: c = 28 / 3

a = 1 / 3 + 28 / 3

a = 29 / 3

b = 9

B)

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();

C)

// Phép toán cộng

PhanSo PhanSo::operator+(const PhanSo &other) const {

    PhanSo result;

    result.tu = this->tu \* other.mau + this->mau \* other.tu;

    result.mau = this->mau \* other.mau;

    return result;

}

// Phép toán gán +=

PhanSo& PhanSo::operator+=(const PhanSo &other) {

    this->tu = this->tu \* other.mau + this->mau \* other.tu;

    this->mau = this->mau \* other.mau;

    return \*this;

}

// Phép toán gán từ int sang PhanSo

PhanSo& PhanSo::operator=(int value) {

    this->tu = value;

    this->mau = 1;

    return \*this;

}

// Phép toán ++ (tăng giá trị) - Tiền tố

PhanSo& PhanSo::operator++() {

    this->tu += this->mau;

    return \*this;

}

// Phép toán ++ (tăng giá trị) - Hậu tố

PhanSo PhanSo::operator++(int) {

    PhanSo temp(\*this);

    this->tu += this->mau;

    return temp;

}

**Câu 2:**

a)

Ý nghĩa và Tầm quan trọng:

Ý nghĩa: Đa hình là tính chất của lập trình hướng đối tượng cho phép một đối tượng được thể hiện dưới nhiều hình thức (kiểu dữ liệu) khác nhau.

Tầm quan trọng: Giúp tăng tính linh hoạt, tái sử dụng mã nguồn, và giảm sự phụ thuộc vào kiểu dữ liệu cụ thể.

B)

- Lớp trừu tượng là một lớp mà không thể tạo được đối tượng cụ thể từ nó. Nó Chứa ít nhất một phương thức thuần ảo mà các lớp dẫn xuất phải triển khai.

- Cài Đặt Trong C++:

Để tạo lớp trừu tượng, cần có ít nhất một phương thức thuần ảo. Phương thức thuần ảo được định nghĩa bằng cách đặt '= 0' ở cuối khai báo phương thức.

**Câu 3:**

A)

Khai báo phương thức `Value(x)` và `Derive(x)` trong lớp `Function` với từ khóa `virtual` và gán giá trị bằng 0 (`= 0`) có ý nghĩa là định nghĩa phương thức ảo thuần túy. Điều này làm cho lớp `Function` trở thành lớp trừu tượng, không thể tạo đối tượng cụ thể từ lớp này. Các lớp dẫn xuất phải triển khai cụ thể cho các phương thức này để trở thành lớp hợp lệ và có thể sử dụng.

B)

#include <cmath>

class Cos : public Function {

public:

    double Value(double x) override {

        return cos(x);

    }

    double Derive(double x) override {

        return -sin(x);

    }

};

C)

double DeriveQuotient(Function\* f, Function\* g, double x) {

    if (f != NULL && g != NULL) {

        double numerator = f->Derive(x) \* g->Value(x) - f->Value(x) \* g->Derive(x);

        double denominator = pow(g->Value(x), 2);

        if (denominator != 0) {

            return numerator / denominator;

        } else {

            return 0;

        }

    }

    return 0;

}

double DeriveComposition(Function\* f, Function\* g, double x) {

    if (f != NULL && g != NULL) {

        return f->Derive(g->Value(x)) \* g->Derive(x);

    }

    return 0;

}

D)

#include <cmath>

class Sin : public Function {

private:

    // Đối tượng duy nhất của lớp Sin

    static Sin\* instance;

    // Hàm tạo ẩn để đảm bảo chỉ có thể tạo đối tượng trong lớp

    Sin() {}

public:

    // Hàm tạo lấy theo mẫu Singleton

    static Sin\* getInstance() {

        if (instance == nullptr) {

            instance = new Sin();

        }

        return instance;

    }

    double Value(double x) override {

        return sin(x);

    }

    double Derive(double x) override {

        return cos(x);

    }

};

// Khởi tạo đối tượng instance

Sin\* Sin::instance = nullptr;