OOP REVIEW

PRESENTATION

GROUP 1

OVERVIEW

1. Chủ đề 01
    - Câu 1
    - Câu 2
    - Câu 3
2. Chủ đề 20
    - Câu 1
    - Câu 2
    - Câu 3

CHỦ ĐỀ 01

CHỦ ĐỀ 01

Câu 1: Phạm vi truy cập và Kế thừa trong C++

a. Phân biệt các phạm vi truy cập private, protected, public và cho ví dụ minh hoạ.

b. Khái niệm về sự kế thừa và những ưu điểm của kế thừa trong lập trình.

Câu a.

1. Thuộc tính public: Thành phần có thuộc tính public có thể truy xuất từ bất kỳ đâu, bao gồm cả từ ngoài lớp.

```cpp
#include <iostream>
using namespace std;

class MyClass {
public:
    int publicVar;
};

int main() {
    MyClass obj;
    obj.publicVar = 10;  // Truy xuat va thay doi bien pulic
    cout << "publicVar = " << obj.publicVar << endl;
    return 0;
}
```

2. Thuộc tính private:

- Các thành viên có phạm vi private chỉ có thể được truy cập từ các hàm thành viên của chính lớp đó.
- Các lớp con hoặc đối tượng bên ngoài không thể truy cập trực tiếp các thành viên private của lớp.

```
class MyClass {
private:
    int myPrivateVar;

public:
    // Ham setter de thay doi gia tri cua myPrivateVar
    void setPrivateVar(int value) {
        myPrivateVar = value;
    }

    // Ham setter de truy xuat gia tri cua myPrivateVar
    int getPrivateVar() {
        return myPrivateVar;
    }
};
```

3. Thuộc tính protected:

- Các thành viên có phạm vi truy cập protected không thể được truy cập từ bên ngoài lớp, nhưng có thể được truy cập trong lớp con (lớp kế thừa).

- Điều này giúp bảo vệ dữ liệu trong lớp cha khỏi việc bị thay đổi từ bên ngoài, nhưng đồng thời cho phép lớp con kế thừa và mở rộng chức năng của lớp cha.

3. Thuộc tính protected:

```
class Base {
protected:
    int protectedVar; // Bien protected

public:
    Base() {
        protectedVar = 10;
    }
};

class Derived : public Base {
public:
    void print() {
        cout << "Giá trị của protectedVar: " << protectedVar << endl; // Truy cap
protectedVar tu lop con
    }
};
```

Câu b.

1. Khái niệm về kế thừa: Kế thừa là cơ chế trong lập trình hướng đối tượng cho phép một lớp con kế thừa các thuộc tính và phương thức của lớp cha. Nó giúp biểu diễn mối quan hệ "đặc biệt hóa – tổng quát hóa" giữa các lớp, với các lớp được tổ chức theo một sơ đồ phân cấp.

2. Lợi ích của kế thừa

- **Xây dựng lớp mới từ lớp cũ**
- **Chia sẻ mã chương trình**
- **Tương thích và chuyển kiểu tự động**

3. Ví dụ minh hoạ:

```cpp
// Lop co so Animal
class Animal {
public:
    void eat() {
        cout << "Animal is eating." << endl;
    }
};

// Lop dan xuat Dog ke thua tu Animal
class Dog : public Animal {
public:
    void bark() {
        cout << "Dog is barking." << endl;
    }
};

int main() {
    Dog myDog;
    myDog.eat();   // Output: Animal is eating.
    myDog.bark();  // Output: Dog is barking.
    return 0;
}
```

CHỦ ĐỀ 01

Câu 2 :

a. Xét đoạn chương trình sau. Cho biết kết quả xuất ra màn hình khi thực thi đoạn chương trình sau. Giải thích ngắn gọn tại sao có kết quả đó.

```
class A {
public:
    A() {
        cout << "Constructing A ";
    }
    ~A() {
        cout << "Destructing A ";
    }
};

class B: public A {
public:
    B() {
        cout << "Constructing B ";
    }
    ~B() {
        cout << "Destructing B ";
    }
};

int main() {
    B b1;
    return 0;
}
```

Câu a.

1. Kết quả:

Constructing A Constructing B Destructing B Destructing A

2. Giải thích lý do:

- **Thứ tự gọi constructor: Khi tạo một đối tượng của lớp dẫn xuất (B), constructor của lớp cơ sở (A) được gọi trước, sau đó mới đến constructor của lớp dẫn xuất (B).**
- **Thứ tự gọi destructor: Khi đối tượng bị hủy, destructor của lớp dẫn xuất (B) được gọi trước, sau đó mới đến destructor của lớp cơ sở (A).**

CHỦ ĐỀ 01

Câu 2 :

b. Cho biết đoạn chương trình trên khi biên dịch có lỗi xảy ra hay không? Nếu có lỗi, hãy chỉ ra các lỗi đó và sửa lỗi để chương trình có thể thực thi được:

```
#include <iostream>
using namespace std;
class A {
private:
    int x;
public:
    A(int t) {
        x = t;
    }
    static void f() {
        cout << x;
    }
    int f2() {
        return x;
    }
};

void main() {
    A a;
    f2(a);
}
```

Câu b.

1. Đoạn chương trình có nhiều lỗi:
   - Hàm main()
   - Khởi tạo đối tượng A a
   - Gọi hàm f2(a)
   - Phương thức static void f()

2. Code sau khi sửa lỗi:

```cpp
#include <iostream>
using namespace std;

class A {
private:`
    int x;
public:
    A(int t) {
        x = t;
    }
    void f() {  // Khong phai static
        cout << x;
    }
    int f2() {
        return x;
    }
};

int main() {
    A a(10);  // Khoi tao doi tuong voi tham so
    a.f2();   // Goi phuong thuoc f2()
    return 0;  // Tra ve 0 cho ham main
}
```

CHỦ ĐỀ 01

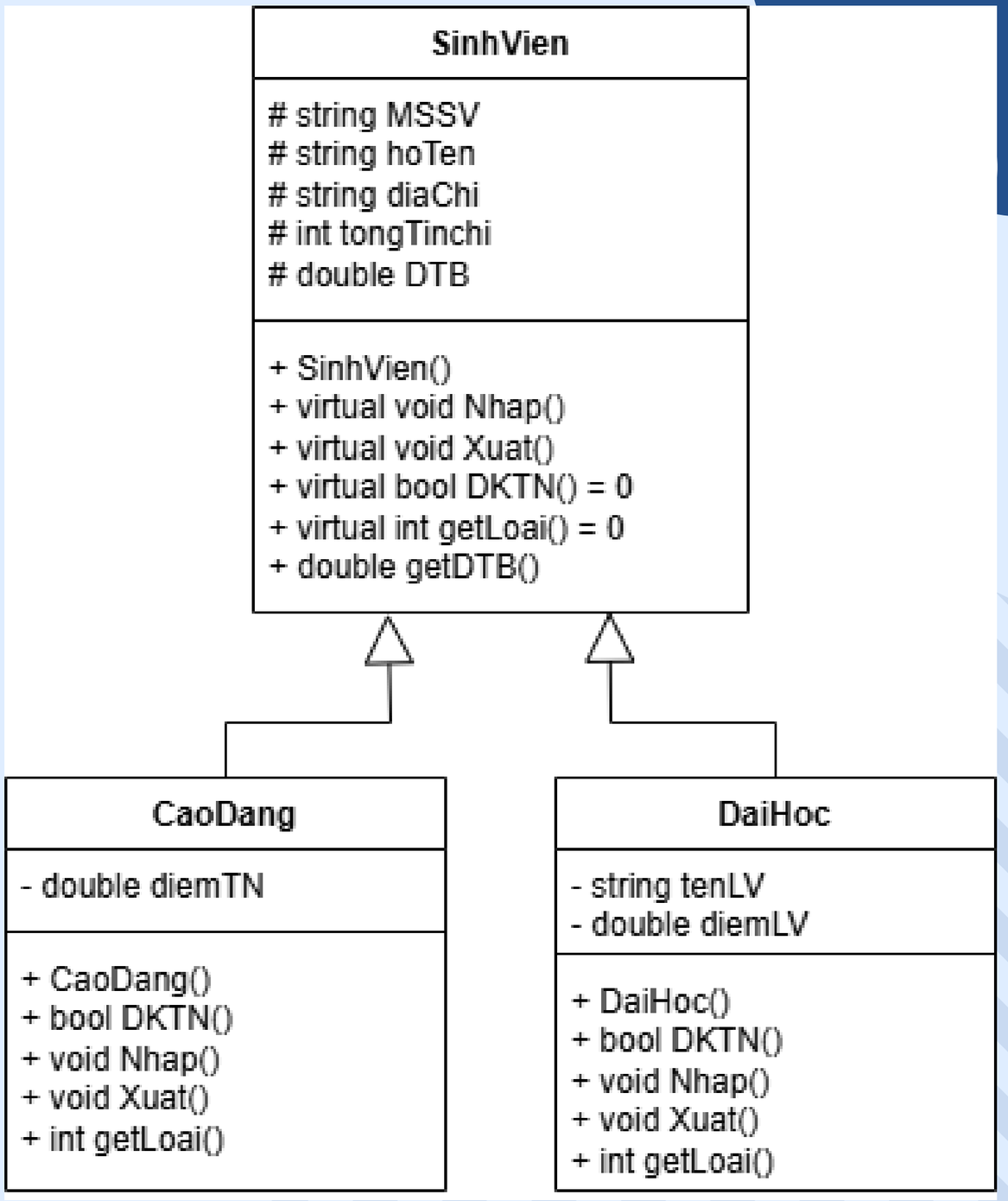
Câu 3:

- Quản lý sinh viên Trường ĐH CNTT TP.HCM với 2 hệ đào tạo: Cao đẳng và Đại học.
- Thông tin cần quản lý:
  - Cao đẳng
  - Đại học
- Điều kiện tốt nghiệp:
  - Cao đẳng: Tín chỉ ≥ 120, DTB ≥ 5, điểm thi tốt nghiệp ≥ 5.
  - Đại học: Tín chỉ ≥ 170, DTB ≥ 5, điểm luận văn ≥ 5.
- Yêu cầu chương trình:
  - Nhập danh sách sinh viên, xác định số lượng sinh viên đủ điều kiện tốt nghiệp.
  - Tìm sinh viên đại học có điểm trung bình cao nhất.

Câu 3:

Thiết kế sơ đồ lớp:

- **Lớp cơ sơ: Sinh Viên**
- **Lớp dẫn xuất: Cao Đẳng, Đại Học**



Câu 3:

Mô tả hướng giải quyết:

- **File sinhvien.h, caodang.h, daihoc.h: Khai báo lớp cơ sở SinhVien và các lớp dẫn xuất CaoDang, DaiHoc với các thuộc tính và phương thức:**
  - **Lớp SinhVien:**
    - **Thuộc tính: MSSV, hoTen, diaChi, tongTinchi, DTB.**
    - **Phương thức: khởi tạo mặc định, nhập/xuất thông tin, kiểm tra điều kiện tốt nghiệp, lấy điểm trung bình.**
  - **Lớp CaoDang:**
    - **Thuộc tính bổ sung: diemTN.**
    - **Phương thức: nhập/xuất thông tin, kiểm tra điều kiện tốt nghiệp.**
  - **Lớp DaiHoc:**
    - **Thuộc tính bổ sung: tenLV, diemLV.**
    - **Phương thức: nhập/xuất thông tin, kiểm tra điều kiện tốt nghiệp.**

- **Khai báo lớp cơ sở Sinh Viên:**

```
// Lop co so SinhVien
class SinhVien {
 protected:
    string MSSV;        // Ma so sinh vien
    string hoTen;       // Ho ten sinh vien
    string diaChi;      // Dia chi sinh vien
    int tongTinchi;     // Tong so tin chi
    double DTB;         // Diem trung binh
 public:
    SinhVien();                    // Constructor mac dinh
    virtual void Nhap();           // Phuong thuc ao de nhap thong tin
    virtual void Xuat();           // Phuong thuc ao de xuat thong tin
    virtual bool DKTN() = 0;       // Phuong thuc ao thuan tuy de kiem tra dieu kien tot nghiep
    virtual int getLoai() = 0;     // Phuong thuc ao thuan tuy de lay loai sinh vien
    double getDTB();               // Phuong thuc lay diem trung binh
};
```

- **Khai báo lớp dẫn xuất Cao Đẳng:**

```
# include "sinhvien.h"
// Lop CaoDang ke thua lop SinhVien
class CaoDang : public SinhVien {
 private:
    double diemTN;       // Diem thi tot nghiep cua sinh vien cao dang
 public:
    CaoDang();           // Constructor mac dinh
    bool DKTN();         // Phuong thuc kiem tra dieu kien tot nghiep cho sinh vien cao dang
    void Nhap();         // Phuong thuc nhap thong tin sinh vien cao dang
    void Xuat();         // Phuong thuc xuat thong tin sinh vien cao dang
    int getLoai();       // Phuong thuc tra ve loai sinh vien (loai cao dang)
};
```

- **Khai báo lớp dẫn xuất Đại Học:**

```
# include "sinhvien.h"
// Lop DaiHoc ke thua lop SinhVien
class DaiHoc : public SinhVien {
 private:
    string tenLV;       // Ten luan van
    double diemLV;      // Diem luan van
 public:
    DaiHoc();           // Constructor mac dinh
    bool DKTN();        // Phuong thuc kiem tra dieu kien tot nghiep cho sinh vien dai hoc
    void Nhap();        // Phuong thuc nhap thong tin sinh vien dai hoc
    void Xuat();        // Phuong thuc xuat thong tin sinh vien dai hoc
    int getLoai();      // Phuong thuc tra ve loai sinh vien (loai dai hoc)
};
```

Câu 3:

Mô tả hướng giải quyết:

- File sinhvien.cpp, caodang.cpp, daihoc.cpp: Định nghĩa các phương thức:
  - Lớp SinhVien:
    - Khởi tạo, nhập/xuất thông tin, lấy điểm trung bình, phương thức ảo kiểm tra điều kiện tốt nghiệp.
  - Lớp CaoDang và DaiHoc:
    - Nhập/xuất thông tin, kiểm tra điều kiện tốt nghiệp, lấy loại sinh viên.

- **Định nghĩa lớp Cao Đẳng:**

```
# include "caodang.h"
CaoDang::CaoDang() {}
bool CaoDang::DKTN() {
    if (tongTinchi >= 120 && DTB >= 5 && diemTN >= 5) return true;
    return false;
}
void CaoDang::Nhap() {
    SinhVien::Nhap();
    cout << "Nhap diem thi tot nghiep: ";
    cin >> diemTN;
}
void CaoDang::Xuat() {
    SinhVien::Xuat();
    cout << "Diem thi tot nghiep: " << diemTN << endl;
}
int CaoDang::getLoai() {
    return 1;
}
```

- **Định nghĩa lớp Đại Học:**

```
# include "daihoc.h"
DaiHoc::DaiHoc() {}
bool DaiHoc::DKTN() {
    if (tongTinchi >= 170 && DTB >= 5 && diemLV >= 5) return true;
    return false;
}
void DaiHoc::Nhap() {
    SinhVien::Nhap();
    cin.ignore();
    cout << "Nhap ten luan van: "; getline(cin, tenLV);
    cout << "Nhap diem luan van: "; cin >> diemLV;
}
void DaiHoc::Xuat() {
    SinhVien::Xuat();
    cout << "Ten luan van: " << tenLV << endl;
    cout << "Diem luan van: " << diemLV << endl;
}
int DaiHoc::getLoai() {
    return 2;
}
```

Câu 3:

Mô tả hướng giải quyết:

- **File main.cpp:**
  - **Nhập số lượng sinh viên.**
  - **Sử dụng vector chứa con trỏ SinhVien để quản lý cả sinh viên cao đẳng và đại học nhờ cơ chế đa hình.**
  - **Nhập danh sách sinh viên, kiểm tra điều kiện tốt nghiệp**
  - **Đếm số sinh viên đủ điều kiện tốt nghiệp.**
  - **Tìm sinh viên đại học có điểm trung bình cao nhất và xuất thông tin tất cả sinh viên có điểm trung bình cao nhất.**
- **Xuất kết quả:**
  - **Số lượng sinh viên đủ điều kiện tốt nghiệp.**
  - **Danh sách thông tin sinh viên đại học có điểm trung bình cao nhất.**

- **Nhập danh sách sinh viên**

```
# include "sinhvien.h"
# include "caodang.h"
# include "daihoc.h"
int main() {
    int n;
    cout << "Nhap so luong sinh vien: "; cin >> n;
    vector<SinhVien*> dssv(n);
    bool coSVDH = false;
    for (int i = 0; i < n; i++) {
        int loai;
        cout << "Nhap loai sinh vien (1: Cao Dang, 2: Dai Hoc): "; cin >> loai;
        SinhVien *sv;
        if (loai == 1)
            sv = new CaoDang();
        else if (loai == 2) {
            sv = new DaiHoc();
            coSVDH = true;
        }
        sv->Nhap();
        dssv[i] = sv;
    }
```

- **Đếm số lượng sinh viên đủ điều kiện tốt nghiệp**

```
int cnt = 0;
for (auto &sv : dssv)
    if (sv->DKTN()) cnt++;
cout << "So luong sinh vien du dieu kien tot nghiep: " << cnt << endl;
```

- **Tìm sinh viên Đại học có điểm trung bình cao nhất**

```
double maxDTB = 0;
if (coSVDH) {
    for (auto &sv: dssv)
        if (sv->getLoai() == 2 && sv->getDTB() > maxDTB)
            maxDTB = sv->getDTB();
    cout << "Sinh vien dai hoc co diem trung binh cao nhat:\n";
    for (auto &sv: dssv)
        if (sv->getLoai() == 2 && sv->getDTB() == maxDTB)
            sv->Xuat();
}
else cout << "Khong co sinh vien dai hoc trong danh sach.\n";
```

CHỦ ĐỀ 20

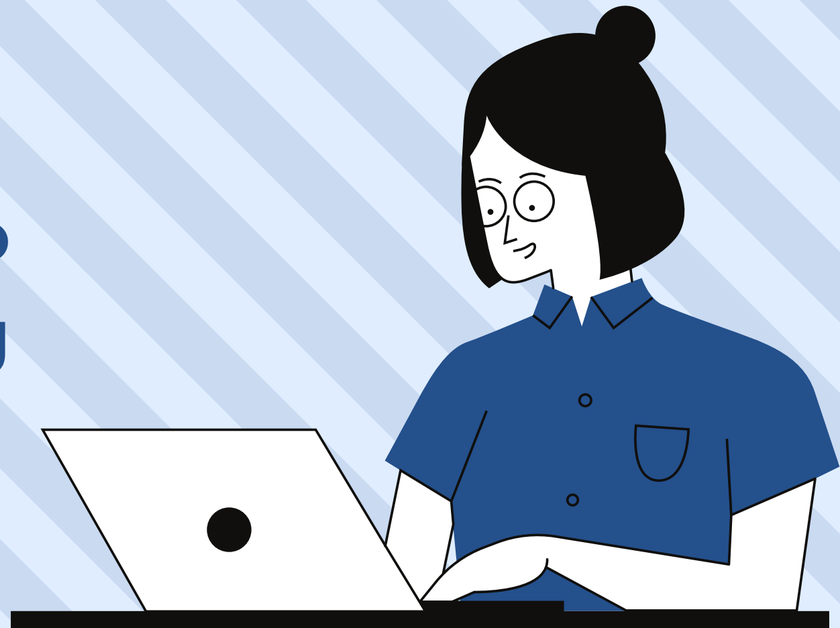
CHỦ ĐỀ 20

Câu 1:

Câu b.

- **Phân biệt các phạm vi truy cập private, protected và public trong một lớp. Cho ví dụ về cách thức từ bên ngoài một lớp muốn truy cập các thành phần được thiết lập private trong lớp đó**

- **Trình bày phương thức ảo là gì? Hãy cho biết trường hợp nào cần sử dụng phương thức này. Cho ví dụ minh hoạ.**



1. Phân biệt các phạm vi truy cập:

- **Thuộc tính public: Thành phần có thuộc tính public có thể truy xuất từ bất kỳ đâu, bao gồm cả từ ngoài lớp.**
- **Thuộc tính private:**
  - **Các thành viên có phạm vi private chỉ có thể được truy cập từ các hàm thành viên của chính lớp đó.**
  - **Các lớp con hoặc đối tượng bên ngoài không thể truy cập trực tiếp các thành viên private của lớp.**
- **Thuộc tính protected:**
  - **Các thành viên có phạm vi truy cập protected không thể được truy cập từ bên ngoài lớp, nhưng có thể được truy cập trong lớp con (lớp kế thừa).**

1. Ví dụ về cách thức từ bên ngoài một lớp muốn truy cập các thành phần được thiết lập private trong lớp đó

- **Sử dụng phương thức truy vấn:**

```
class Nguoi {
private :
    string ten ;
public :
    Nguoi ( string name ) : ten ( name ) {}
    // Phuong thuc truy van truy cap thuoc tinh ten
    string getTen () {
      return ten ;
    }
};

int main () {
    Nguoi nguoi ( " Dang Quoc Cuong Dep Trai Nhat Vu Tru " ) ;
    // Truy cap cac thanh vien private thong qua cac ham cong khai
    cout << nguoi . getTen () ;
    return 0;
}
```

1. Ví dụ về cách thức từ bên ngoài một lớp muốn truy cập các thành phần được thiết lập private trong lớp đó

- **Sử dụng hàm bạn:**

```
class Nguoi {
private :
    string ten ;
public :
    Nguoi ( string name ) : ten ( name ) {}
    // Su dung ham ban
    friend void Xuat ( Nguoi & nguoi ) ;
};

void Xuat ( Nguoi & nguoi ) {
    cout << nguoi.ten ;
}

int main () {
    Nguoi nguoi ( " Cap Kim Hai Anh " ) ;
    // Truy cap truc tiep thong qua ham ban
    Xuat ( nguoi ) ;
    return 0;
}
```

2. Khái niệm phương thức ảo:

- Là phương thức của lớp cơ sở có thể được ghi đè trong các lớp dẫn xuất, là cách thể hiện tính đa hình trong ngôn ngữ C++.
- Trường hợp cần sử dụng phương thức ảo:
  - Các phương thức ở lớp cơ sở có tính đa hình phải được định nghĩa là một phương thức ảo.
  - Khi muốn đảm bảo rằng phương thức phù hợp trong lớp dẫn xuất được gọi ngay cả khi sử dụng con trỏ hoặc tham chiếu kiểu lớp cơ sở.

2. Ví dụ minh hoạ:

Không sử dụng phương thức ảo:

```
class Animal {
public:
    void speak() {
        cout << "Animal speaks" << endl;
    }
};

class Dog : public Animal {
public:
    void speak() {
        cout << "Dog barks" << endl;
    }
};

int main() {
    Animal* animal = new Dog();
    animal->speak();
    return 0;
}
```

Kết quả: `Animal speaks`

Sử dụng phương thức ảo:

```
class Animal {
public:
    virtual void speak() {
        cout << "Animal speaks" << endl;
    }
};

class Dog : public Animal {
public:
    void speak() override {
        cout << "Dog barks" << endl;
    }
};

int main() {
    Animal* animal = new Dog();
    animal->speak();
    return 0;
}
}
```

Kết quả: `Dog barks`

CHỦ ĐỀ 20

Câu 2:

- Đề bài yêu cầu xây dựng chương trình quản lý thời gian hoạt động của một robot.
- Thời gian hoạt động của robot được tính bằng hiệu giữa thời gian bắt đầu và kết thúc. Cần thiết kế hai lớp:
  - CDurationSpan: Lớp biểu diễn khoảng thời gian với ba thuộc tính: giờ, phút, giây.
  - CRobot: Lớp đại diện cho robot, bao gồm hai đối tượng CDurationSpan: thời gian bắt đầu và thời gian kết thúc. Cần có phương thức tính thời gian hoạt động của robot.
- Yêu cầu: Định nghĩa các lớp CDurationSpan và lớp CRobot thích hợp để chương trình bên dưới không bị biên dịch lỗi và chạy đúng.

CHỦ ĐỀ 20

Câu 2:

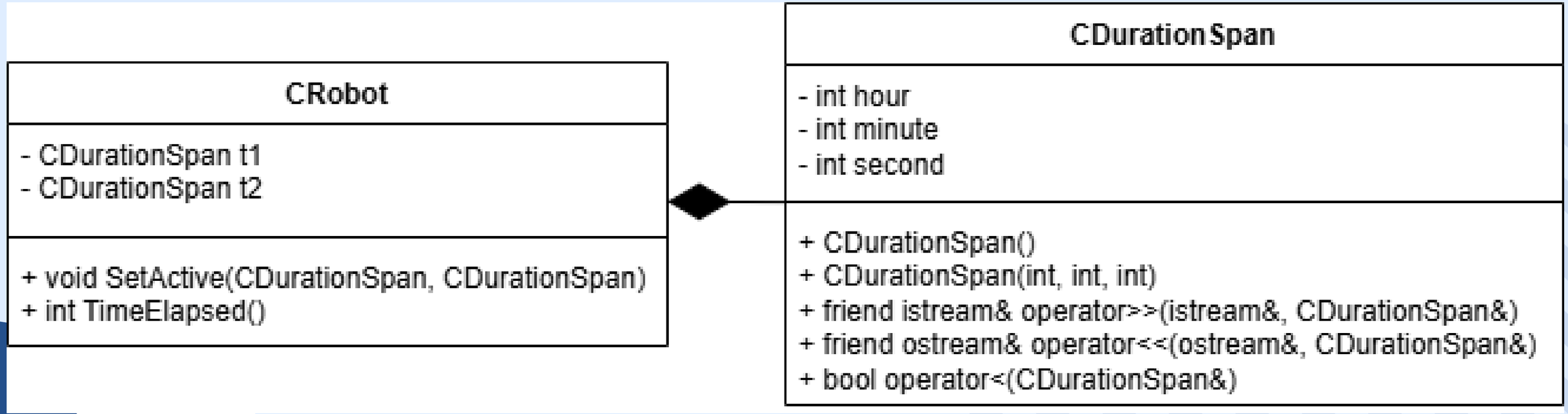
Chương trình đề bài cho:

```
int main() {
    CDurationSpan t1();
    CDurationSpan t2(23, 55, 15);
    CRobot r;

    cin >> t1;
    cin >> t2;

    if (t1 < t2) {
        r.SetActive(t1, t2);
        cout << "Thoi gian Robot hoat dong la: " << r.TimeElapsed();
    }
    else
    {
        cout << "Thiet lap thoi gian khong hop le!";
    }
}
```

Câu 2:

Thiết kế sơ đồ lớp:

- **Lớp RoBot**
- **Lớp DurationSpan**



Câu 2:

Mô tả hướng giải quyết:

File CDurationSpan.h:

- **Khai báo lớp CDurationSpan với các thuộc tính: giờ, phút, giây. Cung cấp các toán tử nhập và xuất.**
- **Cung cấp phương thức so sánh thời gian (operator<) để so sánh hai đối tượng CDurationSpan.**

File CRobot.h:

- **Khai báo lớp CRobot với hai đối tượng CDurationSpan: thời gian bắt đầu và kết thúc.**
- **Phương thức SetActive để thiết lập thời gian bắt đầu và kết thúc cho robot.**
- **Phương thức TimeElapsed để tính toán thời gian hoạt động của robot.**

- **Khai báo lớp DurationSpan**

```
# include<bits/stdc++.h>
# pragma once
using namespace std;


class CDurationSpan {
private:
    int hour, minute, second;
    friend class CRobot;
public:
    CDurationSpan();
    CDurationSpan(int h, int m, int s);
    friend istream& operator >> (istream& is, CDurationSpan& d);
    friend ostream& operator << (ostream& os, const CDurationSpan& d);
    bool operator < (const CDurationSpan& d);
};
```

- **Khai báo lớp Robot**

```
# include<bits/stdc++.h>
# include "CDurationSpan.h"
# pragma once
using namespace std;


class CRobot {
private:
    CDurationSpan t1; // Thoi gian bat dau
    CDurationSpan t2; // Thoi gian ket thuc
public:
    void SetActive (CDurationSpan a, CDurationSpan b);
    int TimeElapsed();
};
```

• Định nghĩa lớp DurationSpan

```
CDurationSpan::CDurationSpan() : hour(0), minute(0), second(0) {}
CDurationSpan::CDurationSpan(int h = 0, int m = 0, int s = 0) : hour(h), minute(m), second(s) {}
istream& operator >> (istream& is, CDurationSpan& d) {
    is >> d.hour >> d.minute >> d.second;
    return is;
}
ostream& operator << (ostream& os, const CDurationSpan& d) {
    os << d.hour << " " << d.minute << " " << d.second;
    return os;
}
bool CDurationSpan::operator < (const CDurationSpan& d) {
    if (hour < d.hour) return true;
    if (hour == d.hour && minute < d.minute) return true;
    if (hour == d.hour && minute == d.minute && second < d.second) return true;
    return false;
}
```

- **Định nghĩa lớp Robot**

```
void CRobot::SetActive (CDurationSpan a, CDurationSpan b) {
    t1 = a;
    t2 = b;
}

int CRobot::TimeElapsed() {
    return (t2.hour - t1.hour) * 3600 + (t2.minute - t1.minute) * 60 + (t2.second - t1.second);
}
```

Câu 2:

Mô tả hướng giải quyết:

File main.cpp:

- **Nhập vào thời gian bắt đầu và kết thúc cho robot.**
- **Kiểm tra nếu thời gian kết thúc nhỏ hơn thời gian bắt đầu, thông báo lỗi.**
- **Nếu hợp lệ, tính và xuất thời gian hoạt động của robot**

- **Hàm main**

```
# include "CDurationSpan.h"
# include "CRobot.h"
using namespace std;

int main() {
    CDurationSpan t1;
    CDurationSpan t2(23, 55, 15);
    CRobot r;
    cin >> t1 >> t2;
    if (t1 < t2) {
        r.SetActive(t1, t2);
        cout << "Thoi gian Robot hoat dong la: " << r.TimeElapsed();
    }
    else {
        cout << "Thiet lap thoi gian khong hop le!";
    }
}
```

CHỦ ĐỀ 20

Câu 3: Quản lý thông tin xe

- Các loại xe:
  - Xe xăng, Xe lai xăng điện, Xe điện.
- Các thuộc tính chung của xe:
  - Số khung, số máy, dung tích, màu sắc, tên hãng sản xuất, số chỗ ngồi, năm sản xuất, xuất xứ, giá nhập khẩu.
- Các thành phần của chi phí chung:
  - Giá xe nhập khẩu: Là giá của chiếc xe được nhập và vận chuyển về đến kho của công ty.
  - Thuế nhập khẩu:
    Công thức tính: Thuế nhập khẩu = 70% x Giá xe nhập khẩu

CHỦ ĐỀ 20

Câu 3: Quản lý thông tin xe

- Thuế tiêu thụ đặc biệt (được tính theo công thức riêng cho từng loại xe):
  - Xe xăng: Thuế tiêu thụ đặc biệt = 60% x (Giá xe nhập khẩu + Thuế nhập khẩu)
  - Xe điện: Thuế tiêu thụ đặc biệt = 50% x (Giá xe nhập khẩu + Thuế nhập khẩu)
  - Xe xăng lai điện: Thuế tiêu thụ đặc biệt = 55% x (Giá xe nhập khẩu + Thuế nhập khẩu)
- Biên lợi nhuận: Công ty để biên lợi nhuận 20% của (Giá xe nhập khẩu + Thuế nhập khẩu + Thuế tiêu thụ đặc biệt)

CHỦ ĐỀ 20

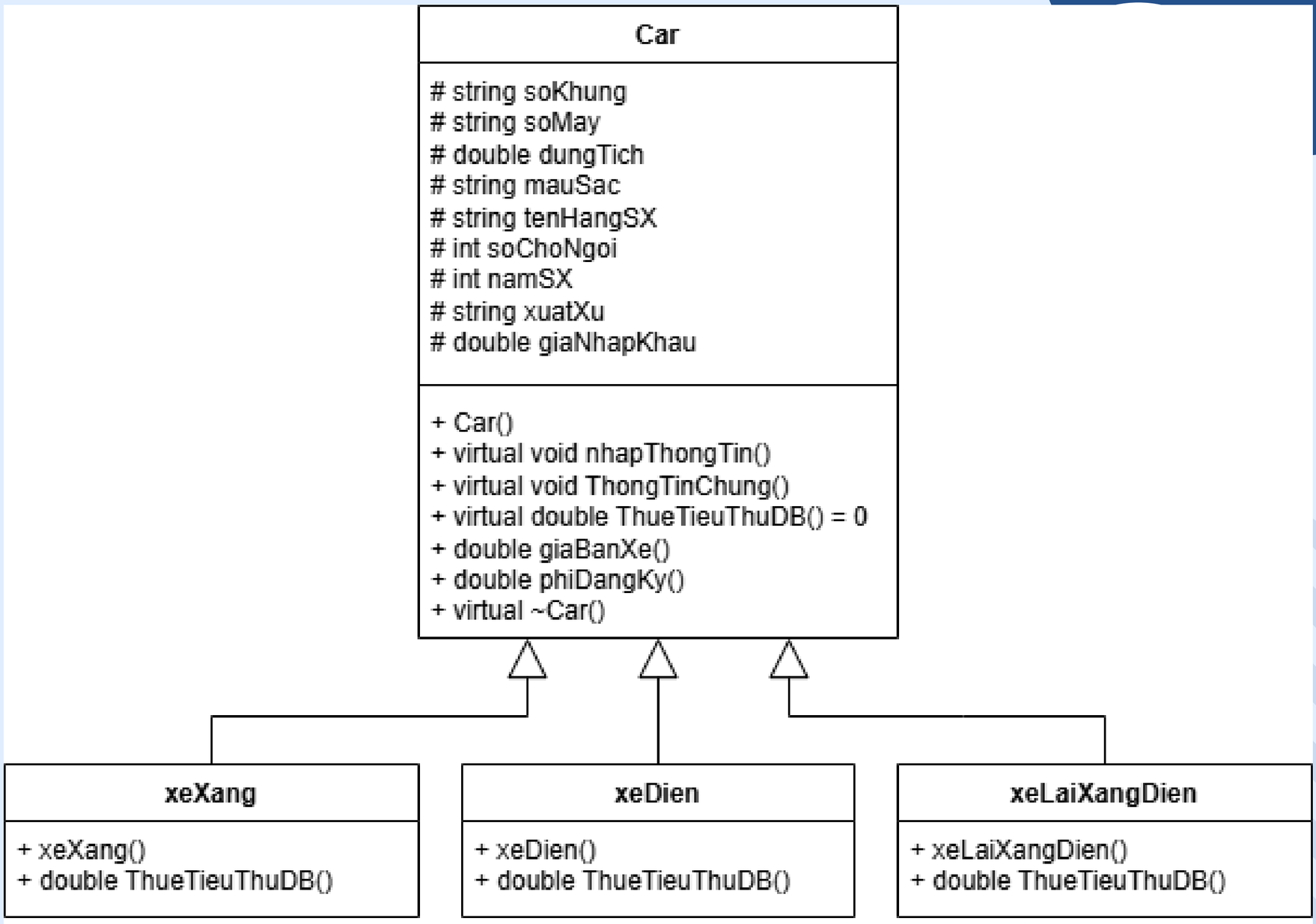
Câu 3: Quản lý thông tin xe

- Giá xe:
  - Công thức tính: Giá xe = Giá xe nhập khẩu + Thuế nhập khẩu + Thuế tiêu thụ đặc biệt + Biên lợi nhuận
  - Thuế VAT: Công thức tính: Thuế VAT = 10% x Giá xe
- Giá bán: Công thức tính: Giá bán = Giá xe + Thuế VAT
- Phí đăng ký:
  - Khi khách hàng mua xe về để sử dụng, cần phải thêm một khoản phí gọi là chi phí đăng ký.
  - Chi phí đăng ký: Công thức tính: Chi phí đăng ký = 2% x Giá xe
- Yêu cầu:
  - Xuất ra thông tin tổng số tiền của tất cả các xe.
  - Xuất ra thông tin của xe có giá bán cao nhất.

Câu 3:

Thiết kế sơ đồ lớp:

- **Lớp cơ sơ: Car**
- **Lớp dẫn xuất: xeXang, xeDien, xeLaiXangDien**



Câu 3:

Mô tả hướng giải quyết:

- **Lớp cơ sở Car:**
  - **Thuộc tính: số khung, số máy, dung tích, màu sắc, tên hãng sản xuất, số chỗ ngồi, năm sản xuất, xuất xứ và giá nhập khẩu.**
  - **Phương thức nhập thông tin, tính giá bán xe, tính phí đăng ký và thuế tiêu thụ đặc biệt.**
  - **Phương thức ThongTinChung() để xuất thông tin chi tiết của xe.**
- **Các lớp dẫn xuất xe con (xeDien, xeLaiXangDien, xeXang):**
  - **Các lớp này kế thừa từ lớp Car và định nghĩa phương thức tính thuế tiêu thụ đặc biệt riêng cho từng loại xe.**

- **Khai báo lớp Car**

```
class Car {
protected:
    string soKhung;
    string soMay;
    double dungTich;
    string mauSac;
    string tenHangSX;
    int soChoNgoi;
    int namSX;
    string xuatXu;
    double giaNhapKhau;
public:
    Car();
    virtual void nhapThongTin();
    virtual double ThueTieuThuDB() = 0;
    double giaBanXe();
    double phiDangKy();
    virtual void ThongTinChung();
    virtual ~Car();
};
```

- **Khai báo lớp Xe Điện**

```
class xeDien : public Car {
public:
    xeDien();
    double ThueTieuThuDB();
};
```

- **Khai báo lớp Xe Xăng**

```
class xeXang : public Car {
public:
    xeXang();
    double ThueTieuThuDB();
};
```

- **Khai báo lớp Xe Lai Xăng Điện**

```
class xeLaiXangDien : public Car {
public:
    xeLaiXangDien();
    double ThueTieuThuDB();
};
```

- **Định nghĩa lớp Xe Điện**

```
xeDien::xeDien() {}

double xeDien::ThueTieuThuDB() {

    return 0.5 * (giaNhapKhau + 0.7 * giaNhapKhau);

}
```

- **Định nghĩa lớp Xe Xăng**

```
xeXang::xeXang() {}

double xeXang::ThueTieuThuDB() {

    return 0.6 * (giaNhapKhau + 0.7 * giaNhapKhau);

}
```

- **Định nghĩa lớp Xe Lai Xăng Điện**

```
xeLaiXangDien::xeLaiXangDien() {}

double xeLaiXangDien::ThueTieuThuDB() {

    return 0.55 * (giaNhapKhau + 0.7 * giaNhapKhau);

}
```

Câu 3:

Mô tả hướng giải quyết:

- File main.cpp:
  - Nhập vào số lượng xe và loại xe.
  - Nhập thông tin chi tiết cho từng loại xe (xe xăng, xe lai xăng điện, xe điện).
  - Tính giá bán và phí đăng ký cho từng xe.
  - Tính tổng số tiền của tất cả các xe và tìm xe có giá bán cao nhất.

- **Nhập danh sách các loại xe**

```
int main() {
    int n;
    cout << "Nhap so luong xe: "; cin >> n;
    vector<Car*> cars;
    for (int i = 1; i <= n; i++) {
        cout << "Nhap thong tin xe thu " << i << '\n';
        int loai; cin >> loai;
        if (loai == 1)
            xeXang* a = new xeXang();
        else if (loai == 2)
            xeLaiXangDien* a = new xeLaiXangDien();
        else xeDien* a = new xeDien();
        a->nhapThongTin();
        cout << a->giaBanXe() << '\n';
        cars.push_back(a);
    }
```

- **Tính tổng số tiền tất cả xe và tìm xe có giá bán cao nhất**

```
double tongSoTien = 0.0;
for (auto it : cars)
    tongSoTien += it->giaBanXe();
cout << "Tong so tien: " << tongSoTien << '\n';
Car* maxCar = nullptr;
for (auto it : cars)
    if (maxCar == nullptr || maxCar->giaBanXe() < it->giaBanXe())
        maxCar = it;
if (maxCar != nullptr) {
    cout << "Thong tin xe co gia cao nhat: " << '\n';
    maxCar->ThongTinChung();
}
```

THANK YOU