# Đề thi cuối kỳ

Môn : Lập trình C nâng cao

Thời gian: 90 phút

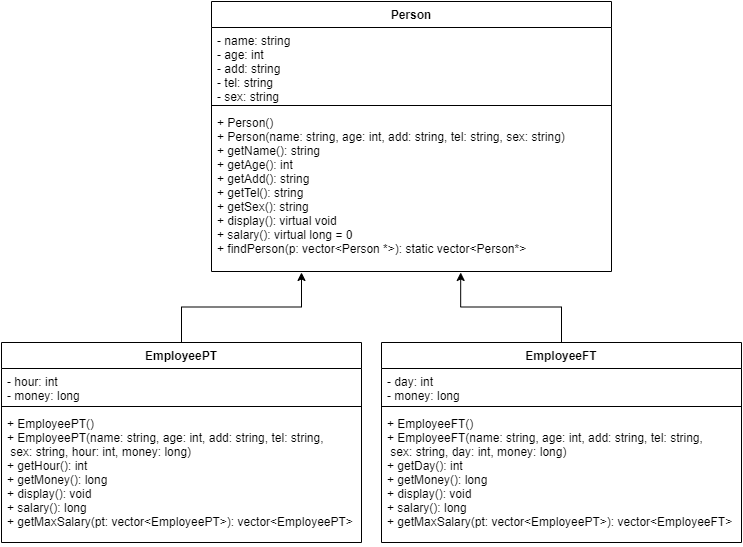
**Bài 1.** Viết hàm nhận một chuỗi S gồm nhiều ký tự bất kỳ, xóa tất cả các ký tự không phải chữ, biến tất cả chữ hoa thành chữ thường và in ra chuỗi kết quả.

Ví dụ 1:

|  |
| --- |
| **Input:**  S = "What IS your NaMe??? My name is &\*what."  **Output:**  what is your name my name is what |

**Bài 2.** Hoàn thiện bài tập lập trình sau:

**Phần 2.1.** Bạn hãy tạo 3 lớp Person, EmployeePT, EmployeeFT với các thông tin sau.



Giải thích:

**Lớp Person**

* name, age, add, tel, sex lần lượt là các thuộc tính chỉ tên người, tuổi, địa chỉ, số điện thoại, giới tính có phạm vi truy cập private.
* Person() là phương thức khởi tạo không tham số.
* Person(name: string, age: int, add: string, tel: string, sex: string) là phương thức khởi tạo 5 tham số.
* display(): virtual void là phương thức hiển thị thông tin Person. Ví dụ name = "Nam", age = 18, sex = "nam", add = "HN", tel = "0123456789" thì sau khi gọi display() màn hình hiển thị:
  + Name: Nam
  + Age: 18
  + Sex: nam
  + Add: HN
  + Tel: 0123456789
* salary(): virtual long = 0 là phương thức thuần ảo tính tiền lương cho nhân viên.
* findPerson(p: vector<Person \*>): static vector<Person\*> là ***phương thức tĩnh*** trả về một vector bao gồm những nhân viên có add = "HN".
* Các phương thức còn lại là getter và setter.

**Lớp EmployeePT**

* Lớp EmployeePT là class kế thừa public từ class Person.
* hour, money lần lượt là số giờ làm việc và số tiền lương/giờ có phạm vi truy cập private.
* EmployeePT() là hàm tạo không tham số.
* EmployeePT(name: string, age: int, add: string, tel: string, sex: string, hour: int, money: long) là hàm tạo 7 tham số.
* display(): void là phương thức hiển thị thông tin EmployeePT. Ví dụ: name = "Nam", age = 18, sex = "nam", add = "HN", tel = "0123456789", hour = 4, money = 1200 thì sau khi gọi display() màn hình hiển thị:
  + Employee Part Time
  + Name: Nam
  + Age: 18
  + Sex: nam
  + Add: HN
  + Tel: 0123456789
  + Hour: 4
  + Money: 1200
  + Salary: 48000
* getMaxSalary(pt: vector<EmployeePT>): vector<EmployeePT> là ***phương thức tĩnh trả*** về một vector bao gồm các nhân viên Part Time có salary nhất.
* Các phương thức còn lại là getter và setter.

**Lớp EmployeeFT**

* Lớp EmployeeFT là class kế thừa public từ class Person.
* day, money lần lượt là số ngày làm việc và số tiền lương/ngày có phạm vi truy cập private.
* EmployeeFT() là hàm tạo không tham số.
* EmployeeFT(name: string, age: int, add: string, tel: string, sex: string, day: int, money: long) là hàm tạo 7 tham số.
* display(): void là phương thức hiển thị thông tin EmployeePT. Ví dụ: name = "Nam", age = 18, sex = "nam", add = "HN", tel = "0123456789", day = 30, money = 300000 thì sau khi gọi display() màn hình hiển thị:
  + Employee Full Time
  + Name: Nam
  + Age: 18
  + Sex: nam
  + Add: HN
  + Tel: 0123456789
  + Day: 30
  + Money: 300000
  + Salary: 9000000
* getMaxSalary(ft: vector<EmployeeFT>): vector<EmployeeFT> là ***phương thức tĩnh trả*** về một vector bao gồm các nhân viên Full Time có salary nhất.
* Các phương thức còn lại là getter và setter

**Phần 2.2.** Viết hàm main để kiểm thử tính chính xác của chương trình.

* Khai báo vector gồm 3 đối tượng lớp EmployeeFT và khởi tạo giá trị cho các đối tượng này thông qua hàm tạo có tham số.
* Hiển thị thông tin của các đối tượng này.
* Hiển thị thông tin của các đối tượng EmployeeFT có lương lớn nhất.
* Khai báo vector gồm 3 đối tượng lớp EmployeePT và khởi tạo giá trị cho các đối tượng này thông qua hàm tạo có tham số.
* Hiển thị thông tin của các đối tượng này.
* Hiển thị thông tin của các đối tượng EmployeeFT có lương lớn nhất.
* Tạo vector các con trỏ Person \* lưu địa chỉ ít nhất 2 đối tượng lớp EmployeeFT và ít nhất 1 đối tượng lớp EmployeePT.
* Sử dụng phương thức findPerson của lớp Person để in ra các nhân viên ở Hà nội sử dụng danh sách con trỏ các đối tượng lớp Person được tạo ở trên.

# 

# Đáp án - Đề thi cuối kỳ

Môn : Lập trình C nâng cao

Thời gian: 90 phút

# 

**Bài 2.**

**Phần 2.1.**

### Person.cpp

#pragma once

#include <iostream>

#include <vector>

using namespace std;

class Person

{

private:

string name;

int age;

string add;

string tel;

string sex;

public:

Person() {}

Person(string name, int age, string add, string tel, string sex)

{

this->name = name;

this->age = age;

this->add = add;

this->tel = tel;

this->sex = sex;

}

string getName()

{

return name;

}

int getAge()

{

return age;

}

string getAdd()

{

return add;

}

string getTel()

{

return tel;

}

string getSex()

{

return sex;

}

virtual void display()

{

cout << "Name: " << name << endl;

cout << "Age: " << age << endl;

cout << "Sex: " << sex << endl;

cout << "Add: " << add << endl;

cout << "Tel: " << tel << endl;

}

virtual long salary() = 0;

static vector<Person \*> findPerson(vector<Person \*> p)

{

vector<Person \*> v;

for (int i = 0; i < p.size(); i++)

{

if (p[i]->add == "HN")

{

v.push\_back(p[i]);

}

}

return v;

}

};

### EmployeeFT.cpp

#include "Person.cpp"

class EmployeeFT : public Person

{

private:

int day;

long money;

public:

EmployeeFT() : Person() {}

EmployeeFT(string name, int age, string add, string tel, string sex, int day, long money) : Person(name, age, add, tel, sex), day(day), money(money) {}

int getDay()

{

return day;

}

long getMoney()

{

return money;

}

long salary()

{

return long(day) \* money;

}

void display()

{

cout << "Employee Full Time" << endl;

Person::display();

cout << "Day: " << day << endl;

cout << "Money: " << money << endl;

cout << "Salary: " << salary() << endl;

}

static vector<EmployeeFT> getMaxSalary(vector<EmployeeFT> ft)

{

long max = ft[0].salary();

for (int i = 1; i < ft.size(); i++)

{

if (ft[i].salary() > max)

{

max = ft[i].salary();

}

}

vector<EmployeeFT> v;

for (int i = 0; i < ft.size(); i++)

{

if (ft[i].salary() == max)

{

v.push\_back(ft[i]);

}

}

return v;

}

};

### EmployeePT.cpp

#include "Person.cpp"

class EmployeePT : public Person

{

private:

int hour;

long money;

public:

EmployeePT() : Person() {}

EmployeePT(string name, int age, string add, string tel, string sex, int hour, long money) : Person(name, age, add, tel, sex), hour(hour), money(money) {}

int getHour()

{

return hour;

}

long getMoney()

{

return money;

}

long salary()

{

return long(hour) \* money;

}

virtual void display()

{

cout << "Employee Part Time" << endl;

Person::display();

cout << "Hour: " << hour << endl;

cout << "Money: " << money << endl;

cout << "Salary: " << salary() << endl;

}

static vector<EmployeePT> getMaxSalary(vector<EmployeePT> pt)

{

long max = pt[0].salary();

for (int i = 1; i < pt.size(); i++)

{

if (pt[i].salary() > max)

{

max = pt[i].salary();

}

}

vector<EmployeePT> v;

for (int i = 0; i < pt.size(); i++)

{

if (pt[i].salary() == max)

{

v.push\_back(pt[i]);

}

}

return v;

}

};

### **Phần 2.2.**

int main()

{

vector<EmployeeFT> fulltimes(3);

fulltimes[0] = EmployeeFT("A", 1, "HN", "123", "M", 10, 1000);

fulltimes[1] = EmployeeFT("B", 2, "HM", "234", "M", 5, 1000);

fulltimes[2] = EmployeeFT("C", 3, "HN", "345", "M", 3, 1000);

for (auto &v : fulltimes)

v.display();

for (auto &v : EmployeeFT::getMaxSalary(fulltimes))

v.display();

vector<EmployeePT> parttimes(3);

parttimes[0] = EmployeePT("D", 1, "HP", "123", "M", 10, 1000);

parttimes[1] = EmployeePT("E", 2, "HN", "234", "M", 5, 1000);

parttimes[2] = EmployeePT("F", 3, "HN", "345", "M", 3, 1000);

for (auto &v : parttimes)

v.display();

for (auto &v : EmployeePT::getMaxSalary(parttimes))

v.display();

vector<Person \*> persons({&fulltimes[0], &fulltimes[1], &parttimes[0]});

for (auto &v : Person::findPerson(persons))

v->display();

}