# Đề thi cuối kỳ

Môn : Lập trình C nâng cao

Thời gian: 90 phút

**Bài 1. Xác thực địa chỉ IP**

Viết chương trình để xác thực địa chỉ IPv4. Địa chỉ IPv4 về mặt quy tắc được biểu diễn bao gồm bốn số nguyên, mỗi số từ 0 đến 255, được phân tách bằng dấu chấm, ví dụ: 172.16.254.1.

Một địa chỉ IPv4 hợp lệ có dạng x1.x2.x3.x4 trong đó 0 <= (x1, x2, x3, x4) <= 255. Do đó, chúng ta có thể viết dạng tổng quát của địa chỉ IPv4 là

(0-255). (0-255). (0-255). (0-255).

**Lưu ý**: Ở đây chúng ta chỉ xem xét các số từ 0 đến 255 và bất kỳ số 0 đứng đầu bổ sung nào sẽ bị coi là không hợp lệ.

**Nhiệm vụ:**

- Hoàn thành hàm isValid trả về 1 nếu địa chỉ IPv4 đã cho là hợp lệ, còn lại trả về 0. Hàm lấy địa chỉ IPv4 làm tham số duy nhất ở dạng chuỗi.

- command line

Ví dụ 1:

|  |
| --- |
| **Input:**  IPv4 address = 222.111.111.111  **Output:** 1  Giải thích: các tiêu chí đều thỏa mãn |

Ví dụ 2:

|  |
| --- |
| **Input:**  IPv4 address = 5555..555  **Output:** 0  Giải thích: 5555..555 thiếu số  **Input:**  IPv4 address = 1.2.3.4.5  **Output:** 0  Giải thích: dãy 1.2.3.4.5 thừa số  **Input:**  IPv4 address = 1.2.3.4.5.  **Output:** 0  Giải thích: dãy 1.2.3.4.5. không đúng format |

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

**Phần 2.1.** Bạn hãy tạo 3 lớp People, Family, Group với các thông tin như phần giải thích bên dưới.

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

* Khai báo vector gồm ít nhất 4 đối tượng lớp People và tạo giá trị cho các đối tượng này thông qua các hàm tạo có tham số. Trong đó có ít nhất 3 đối tượng có job khác rỗng (có việc) và 1 đối tượng có string job là rỗng (thất nghiệp).
* Hiển thị danh sách người vừa khởi tạo và các thông tin tương ứng.
* Khai báo vector gồm các đối tượng Family; vector này chứa ít nhất 2 đối tượng Family sử dụng danh sách People ở trên. Mỗi đối tượng Family sẽ có ít nhất 2 người và có ít nhất 1 đối tượng Family có 1 người thất nghiệp.
* Hiển thị danh sách hộ gia đình trong vector Family vừa tạo là người nghèo sử dụng kết quả trả về của phương thức tĩnh getPoorHousehold của lớp Family.
* Hiển thị những người trong những hộ gia đình cùng họ trong danh sách đối tượng Family ở trên với họ là một giá trị nào đó được sử dụng làm tham số của phương thức findFamilySurename.
* Tạo một đối tượng Group trong đó có ít nhất hai hộ gia đình.
* Hiển thị nội dung của group thông qua phương thức display() của group.
* Hiển thị số hộ gia đình trong group.
* Hiển thị tuổi trung bình của group.
* Hiển thị danh sách những người thất nghiệp trong group.

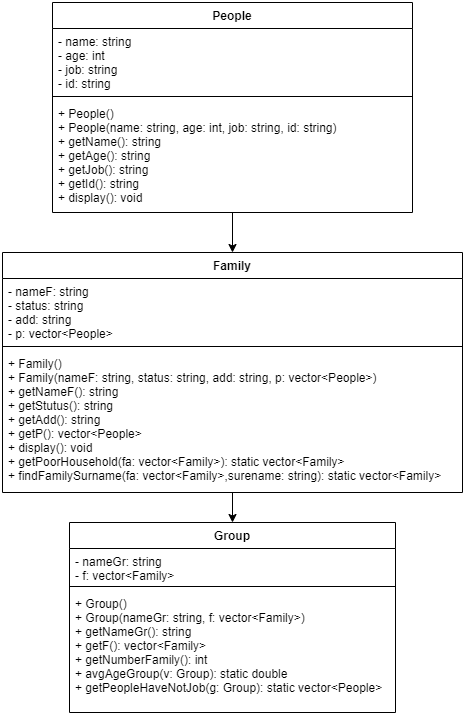
**Giải thích:**

**Lớp People:**

* name, age, job, id lần lượt là các thuộc tính chỉ tên, tuổi, nghề nghiệp, mã định danh của mỗi người có phạm vi truy cập private.
* People() là phương thức khởi tạo không tham số.
* People(name: string, age: int, job: string, id: string) là phương thức khởi tạo 4 tham số.
* display(): void là phương thức hiển thị thông tin People ra màn hình. Ví dụ ta có name = "Nguyen Minh Hieu", age = 20, job = "Dev", id = "1" thì sau khi gọi phương thức display() màn hình hiển thị:
  + Name: Nguyen Minh Hieu
  + Age: 20
  + Job: Dev
  + Id: 1
* Các phương thức còn lại là getter và setter.

**Lớp Family**

* Lớp Family để quản lí các thành viên trong gia đình. Một Family có nhiều People.
* nameF, status, add, p lần lượt là các thuộc tính chỉ tên chủ hộ, thực trạng, địa chỉ, danh sách các thành viên của hộ gia đình có phạm vi truy cập private. Tên chủ hộ được lấy tên của người đầu tiên trong danh sách các thành viên, status có các giá trị là poor, normal, rich tương ứng với hộ nghèo, hộ bình dân, hộ người giàu.
* Family() là phương thức khởi tạo không tham số.
* Family(nameF: string, add: string, status: string, p: vector<People>) là phương thức khởi tạo 4 tham số.
* display(): void là phương thức hiển thị thông tin hộ gia đình ra màn hình. Bạn tham khảo code mẫu bên dưới.
* getPoorHousehold(fa: vector<Family>): static vector<Family> là phương thức tĩnh đầu vào là danh sách các hộ ra đình và đầu ra là danh sách các hộ gia đình đang có status = "poor".
* findFamilySurname(fa: vector<Family>,surename: string): static vector<Family> là phương thức tĩnh đầu vào là danh sách các hộ gia đình và tên của chủ hộ bạn muốn tìm; đầu ra là một danh sách các hộ gia đình có tên chủ hộ giống với surename.
* Các phương thức còn lại là getter và setter.



**Lớp Group**

* Lớp Group dùng để quản lí các hộ gia đình. Một Group có nhiều Family.
* nameGr, f lần lượt là các thuộc tính chỉ tên phường và danh sách các hộ gia đình có phạm vi truy cập private.
* Group() là phương thức khởi tạo không tham số.
* Group(nameGr: string, f: vector<Family>) là phương thức khởi tạo 2 tham số.
* getNumberFamily(): int là phương thức trả về số hộ gia đình của một Group.
* avgAgeGroup(v: Group): static double là phương thức tĩnh có đầu vào là một Group và đầu ra là độ tuổi trung bình của Group đó.
* getPeopleHaveNotJob(g: Group): static vector<People> là phương thức tĩnh đầu vào là một Group đầu ra là một danh sách những người không có công ăn việc làm tức là Job = "".
* Các phương thức còn lại là getter và setter.

# Đáp án - Đề thi cuối kỳ

Môn : Lập trình C nâng cao

Thời gian: 90 phút

# 

**Bài 2.**

**Phần 2.1.**

People.cpp

#pragma once

#include <iostream>

#include <vector>

using namespace std;

class People

{

private:

string name;

int age;

string job;

string id;

public:

People() {}

People(string name, int age, string job, string id) : name(name), age(age), job(job), id(id) {}

string getName()

{

return name;

}

int getAge()

{

return age;

}

string getJob()

{

return job;

}

string getId()

{

return id;

}

void display()

{

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

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

cout << "Job: " << job << endl;

cout << "Id: " << id << endl;

}

};

### Family.cpp

#pragma once

#include "People.cpp"

class Family

{

private:

string nameF;

string status;

string add;

vector<People> p;

public:

Family()

{

}

Family(string add, string status, vector<People> p) : add(add), status(status), p(p)

{

nameF = p[0].getName();

}

string getNameF()

{

return nameF;

}

string getStatus()

{

return status;

}

string getAdd()

{

return add;

}

vector<People> getP()

{

return p;

}

void display()

{

cout << "householder's name: " << nameF << endl;

cout << "Number Member: " << p.size() << endl;

cout << "Address: " << add << endl;

cout << "Status: " << status << endl;

cout << "List Information Member" << endl;

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

{

p[i].display();

}

}

static vector<Family> getPoorHousehold(vector<Family> &fa)

{

vector<Family> v;

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

{

cout << fa[i].getStatus() << endl;

if (fa[i].getStatus() == "poor")

{

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

// cout << fa[i].getNameF() << endl;

}

}

return v;

}

static vector<Family> findFamilySurename(vector<Family> &fa, string surename)

{

vector<Family> v;

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

{

if (fa[i].getNameF() == surename)

{

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

}

}

return v;

}

};

### Group.cpp

#include "Family.cpp"

class Group

{

private:

string nameGr;

vector<Family> f;

public:

Group() {}

Group(string name, vector<Family> &f) : nameGr(name), f(f) {}

void display()

{

cout << "Name Group " << nameGr << endl;

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

{

f[i].display();

}

}

vector<Family> getF()

{

return f;

}

int getNumberFamily()

{

return f.size();

}

static double avgAgeGroup(Group &v)

{

vector<Family> F(v.getF());

double sum = 0;

int n = 0;

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

{

vector<People> P(F[i].getP());

for (int j = 0; j < P.size(); j++)

{

sum += P[j].getAge();

}

n += P.size();

}

return sum / n;

}

static vector<People> getPeopleHaveNotJob(Group &g)

{

vector<People> v;

vector<Family> F(g.getF());

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

{

vector<People> P(F[i].getP());

for (int j = 0; j < P.size(); j++)

{

if (P[j].getJob() == "")

{

v.push\_back(P[j]);

}

}

}

return v;

}

};

### **Phần 2.2.**

int main()

{

vector<People> people(4);

people[0] = People("A", 1, "1", "1");

people[1] = People("B", 2, "2", "1");

people[2] = People("C", 3, "3", "1");

people[3] = People("D", 4, "", "1");

cout << "People:" << endl;

for (auto &v : people)

v.display();

vector<Family> families(2);

families[0] = Family("HN", "poor", {people[0], people[1]});

families[1] = Family("SG", "poor", {people[2], people[3]});

cout << "Poor families:" << endl;

for (auto &v : Family::getPoorHousehold(families))

cout << v.getNameF() << endl;

cout << "Family SurName:" << endl;

for (auto &v : Family::findFamilySurename(families, "A"))

v.display();

cout << "Name group" << endl;

Group groups("1", families);

groups.display();

cout << "Number of families: " << groups.getNumberFamily() << endl;

cout << "Ave age of group: " << groups.avgAgeGroup(groups) << endl;

cout << "Unemployment" << endl;

for (auto &v : Group::getPeopleHaveNotJob(groups))

v.display();

}