# Đề thi cuối kỳ

Môn : Lập trình C nâng cao

Thời gian: 90 phút

**Bài 1.** **Oẳn tù tì 2 người (8 5)**

Có 2 người chơi A và B đang tranh giải vô địch thế giới Oẳn Tù Tì. Mỗi round mỗi người chơi phải quyết định ra hoặc Búa (B), hoặc Bao (G), hoặc Kéo (K). Người thắng mỗi round sẽ được 1 điểm.

Nhiệm vụ:

- Viết hàm getWinner in ra màn hình ai là người thắng. Hàm nhận 2 tham số dạng chuỗi: chuỗi đầu của người chơi A, chuỗi sau của người chơi B. Mỗi ký tự trong chuỗi tương ứng với nước đi của người chơi mỗi round. Ví dụ: "BBG" nghĩa là round 1 và 2 người chơi ra Búa, round 3 ra Bao.

- Hàm trả về lỗi nếu 2 chuỗi tham số không hợp lệ (xem lỗi ở ví dụ bên dưới)

Ví dụ 1:

|  |
| --- |
| **Input:**  BBK GBB  **Output:** B  Giải thích: B thắng do có 2 điểm thắng ở round 1 (G thắng B) và round 3 (B thắng K)  **Input:**  BBKK KKBB  **Output:** Hoa  Giải thích: Trận này hòa, mỗi bên 2 điểm |

Ví dụ 2:

|  |
| --- |
| **Input:**  BBKB GBB  **Output:** Error  Giải thích: Số round không bằng nhau, A 4 round trong khi B chỉ có 3 round |

Ví dụ 3:

|  |
| --- |
| **Input:**  BSA BGK  **Output:** Error  Giải thích: "S" "A" là các ký tự lạ |

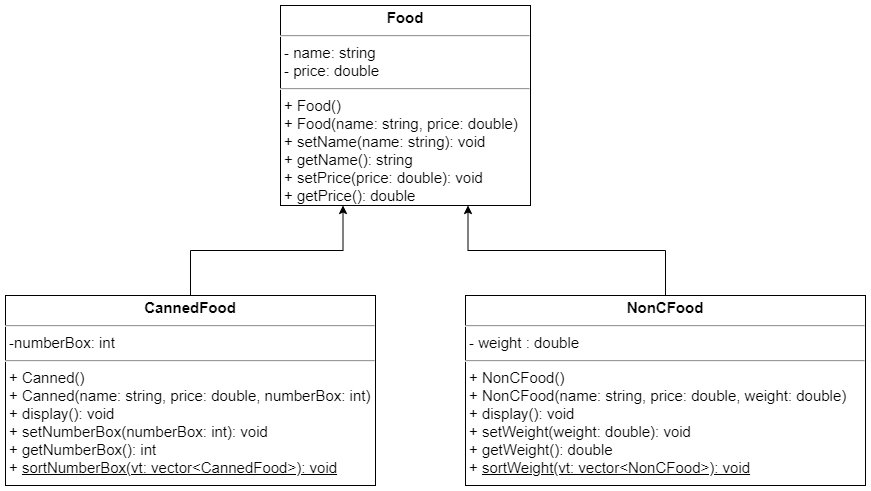
**Bài 2.** Hoàn thiện bài tập lập trình sau:

**Bài 2.1.** Bạn hãy tạo ra 3 lớp Food, CannedFood, NonCFood trong với thông tin như trong phần giải thích.

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

* Khai báo vector gồm 3 đối tượng lớp CannedFood và khởi tạo giá trị cho các đối tượng này thông qua hàm tạo.
* Tìm đối tượng có numberBox lớn nhất và thiết lập giá trị price mới bằng ½ giá trị price cũ của đối tượng này.
* Gọi hàm sortNumberBox để sắp xếp các đối tượng CannedFood ở trên và hiển thị nội dung các đối tượng sau khi sắp xếp.
* Khai báo vector gồm 3 đối tượng lớp NonCFood và khởi tạo giá trị cho các đối tượng này thông qua hàm tạo.
* Tìm đối tượng có weight lớn nhất và thiết lập giá trị weight mới bằng ½ giá trị weight cũ của đối tượng này.
* Gọi hàm sortWeight để sắp xếp các đối tượng NonCFood ở trên và và hiển thị nội dung các đối tượng sau khi sắp xếp.

**Giải thích:**



**Lớp Food**

* name, price là các thuộc tính chỉ tên, giá của thực phẩm và có phạm vi truy cập là private.
* Food() là phương thức tạo mạc định.
* Food(name: string, price: double) là phương thức khởi tạo có tham số dùng để khởi tạo giá trị cho các thuộc tính.
* Các phương thức còn lại là setter và getter.

**Lớp CannedFood**

* Là lớp kế thừa lớp Food chứa thông tin của thực phẩm đóng hộp.
* numberBox là thuộc tính chỉ số lượng hộp.
* CannedFood() là phương thức khởi tạo mạc định.
* CannedFood(name: string, price: double, numberBox: int) là phương thức khởi tạo có tham số dùng để khỏi tạo giá trị cho các thuộc tính.
* display() là phương thức hiển thị ra màn hình thông tin của đối tượng. Ví dụ nếu name = "Ca", price = 1500, numberBox = 10 thì khi gọi phương thức display() màn hình sẽ hiển thị ra:
  + Name: Ca
  + Price: 1500
  + Number box: 10
* sortNumberBox(&vt: vector<CannedFood>) là phương thức tĩnh sắp xếp vector vt theo thứ tự tăng dần của numberBox.
* Các phương thức còn lại là setter và getter.

**Lớp NonCFood**

* Là lớp kế thừa lớp Food chứa thông tin của thực phẩm không đóng hộp.
* weight là thuộc tính chỉ cân nặng của thực phẩm và có phạm vi truy cập là private.
* NonCFood() là phương thức khởi tạo mặc định.
* NonCFood(name: string, price: double, weight: double) là phương thức khởi tạo có tham số dùng để khỏi tạo giá trị cho các thuộc tính.
* display() là phương thức hiển thị ra màn hình thông tin của đối tượng. Ví dụ nếu name = "Ca", price = 1500, weight = 5 thì khi gọi phương thức display() màn hình sẽ hiển thị ra:
  + Name: Ca
  + Price: 1500
  + Weight: 5
* sortWeight(&vt: vector<NonCFood>) là phương thức tĩnh sắp xếp vector vt theo thứ tự tăng dần của weight.
* Các phương thức còn lại là setter và getter.

# 

# Đáp án - Đề thi cuối kỳ

Môn : Lập trình C nâng cao

Thời gian: 90 phút

# 

**Bài 2.**

### Food.cpp

#pragma once

#include <iostream>

using namespace std;

class Food

{

private:

string name;

double price;

public:

Food() {}

Food(string name, double price)

{

this->name = name;

this->price = price;

}

void setName(string name)

{

this->name = name;

}

string getName()

{

return name;

}

void setPrice(double price)

{

this->price = price;

}

double getPrice()

{

return price;

}

};

### CannedFood.cpp

#include "Food.cpp"

#include <vector>

class CannedFood : public Food

{

private:

int numberBox;

public:

CannedFood() {}

CannedFood(string name, double price, int numberBox) : Food(name, price)

{

this->numberBox = numberBox;

}

int getNumberBox()

{

return numberBox;

}

void display()

{

cout << "Name: " << Food::getName() << endl;

cout << "Price: " << Food::getPrice() << endl;

cout << "Number box: " << this->numberBox << endl;

}

static void sortNumberBox(vector<CannedFood> &vt)

{

for (int i = 0; i < vt.size() - 1; i++)

{

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

{

if (vt[j].numberBox < vt[i].numberBox)

{

swap(vt[j], vt[i]);

}

}

}

}

};

### NonCFood.cpp

#include "Food.cpp"

#include <vector>

class NonCFood : public Food

{

private:

double weight;

public:

NonCFood() {}

NonCFood(string name, double price, double weight) : Food(name, price)

{

this->weight = weight;

}

void display()

{

cout << "Name: " << Food::getName() << endl;

cout << "Price: " << Food::getPrice() << endl;

cout << "Weight: " << this->weight << endl;

}

double getWeight()

{

return weight;

}

static void sortWeight(vector<NonCFood> &vt)

{

for (int i = 0; i < vt.size() - 1; i++)

{

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

{

if (vt[j].weight < vt[i].weight)

{

swap(vt[j], vt[i]);

}

}

}

}

};

### Main

int main()

{

vector<CannedFood> cfoods(3);

cfoods[0] = CannedFood("A", 3.0, 3);

cfoods[1] = CannedFood("B", 2.0, 2);

cfoods[2] = CannedFood("C", 1.0, 1);

int maxNumberBoxIndex = 0;

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

if (cfoods[i].getNumberBox() > cfoods[maxNumberBoxIndex].getNumberBox())

maxNumberBoxIndex = i;

cfoods[maxNumberBoxIndex].setPrice(cfoods[maxNumberBoxIndex].getPrice());

CannedFood::sortNumberBox(cfoods);

for (auto v : cfoods)

v.display();

vector<NonCFood> ncfoods(3);

ncfoods[0] = NonCFood("A", 1.0, 1);

ncfoods[1] = NonCFood("B", 2.0, 1);

ncfoods[2] = NonCFood("C", 3.0, 1);

int maxWeightIndex = 0;

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

if (ncfoods[i].getWeight() > ncfoods[maxWeightIndex].getWeight())

maxWeightIndex = i;

ncfoods[maxWeightIndex].setPrice(ncfoods[maxWeightIndex].getPrice());

NonCFood::sortWeight(ncfoods);

for (auto v : ncfoods)

v.display();

}