Bài 1: Class Numbers (Bắt buộc)

Hãy thiết kế một class có tên là Numbers mà có thể được sử dụng để chuyển đổi toàn bộ các khoản tiền nằm trong khoảng từ 0 cho đến 9999 đô la, với các con số này được viết bằng chữ tiếng Anh. Ví dụ, số 713 sẽ được chuyển thành một chuỗi "seven hundred thirteen", và số 8203 sẽ được chuyển thành một chuỗi "eight thousand two hundred three". Class này có một member variable là một số nguyên:

int number;

và một mảng tĩnh các string để lưu trữ các chuỗi được chuyển từ các số tương ứng. Ví dụ, có thể sử dụng một các chuỗi tĩnh như sau nếu sử dụng tiếng Anh:

string lessThan20[20] = {"zero", "one", ..., "eighteen", "nineteen"};

string hundred = "hundred";

string thousand = "thousand";

Class này cần có một cấu tử nhận vào một số nguyên, không âm, và sử dụng nó để khởi tạo đối tượng Numbers. Đồng thời, class cũng cần trang bị một member function có tên là print() để in ra đối tượng Numbers ở dạng chữ. Sau đó, hãy thử chạy demo class này bằng cách viết trong hàm main câu lệnh yêu cầu người dùng nhập vào một số, trong khoảng cho phép, sau đó in số đó ra ở dạng chữ.

Bài 2: Một ngày trong năm (Class DayOfYear)

Giả sử một năm có 365 ngày, hãy viết một class có tên là DayOfYear, sao cho khi khởi tạo đối tượng thuộc class này chỉ cần truyền vào một số nguyên, thể hiện thứ tự của ngày trong năm, và chuyển đổi nó sang thành một chuỗi bao gồm thông tin Tháng, và theo sau đó là ngày thứ mấy của Tháng. Ví dụ:

Ngày thứ 2 của năm sẽ là: Junuary 2.

Ngày thứ 32 của năm sẽ là: February 1.

Ngày thứ 365 của năm sẽ là: December 31.

Cấu tử của class này sẽ nhận vào một tham số là một số nguyên, là ngày của năm, và class có một member function tên là print() để in ra ngày đó theo format Tháng-Ngày như ví dụ trên.

Gợi ý: Class nên có một member variable là một số nguyên để lưu trữ thông tin ngày, và một các member variable khác để lưu trữ các string dùng cho việc chuyển đổi từ số nguyên sang format Tháng-Ngày.

Sau đó, thử tạo đối tượng của class vừa viết với nhiều số thể hiện các ngày khác nhau và in chúng ra để kiểm tra class có hoạt động đúng hay không.

Bài 3: Chỉnh sửa Class DayOfYear

Hãy chỉnh sửa cho class DayOfYear đã viết trong Bài 2, cho phép thêm cấu tử nhận vào 2 tham số, bao gồm: một string thể hiện Tháng, và một số nguyên trong khoảng từ 0 đến 31 thể hiện ngày của Tháng đó. Cấu tử sẽ khởi tạo giá trị biến số nguyên của class để thể hiện cho Tháng và Ngày được truyền vào. Cấu tử sẽ kết thúc chương trình với thông báo lỗi nếu số được nhập vào cấu tử nằm ngoài khoảng ngày cho phép của một tháng (ngày không hợp lệ).

Thêm vào class các hàm quá tải toán tử sau:

- Toán tử tăng lên ++ tiền tố và hậu tố. Các toán tử này thay đổi đối tượng DayOfYear để biểu diễn cho ngày tiếp theo. Nếu ngày hiện tại là ngày cuối cùng của năm thì giá trị của ngày mới sẽ quay trở lại biểu diễn ngày đầu tiên của năm.
- Toán từ giảm xuống -- tiền tố và hậu tố. Các toán tử này thay đổi đối tượng DayOfYear để biểu diễn cho ngày trước đó. Nếu ngày hiện tại là ngày đầu tiên của năm thì giá trị của ngày mới sẽ biểu diễn cho ngày cuối cùng của năm.

Bài 4: Class NumDays

Hãy thiết kế một class có tên là NumDays. Mục đích của class này là lưu trữ một giá trị thể hiện một số giờ làm việc và chuyển đổi chúng sang thành đơn vị ngày công. Ví dụ, 8 giờ làm việc sẽ được chuyển đổi thành 1 ngày công, 12 giờ làm việc sẽ được chuyển đổi thành 1.5 ngày công, và 18 giờ làm việc có thể chuyển đổi thành 2.25 ngày công.

Class cần có một cấu tử nhận vào một số thể hiện giờ làm; cần có các member function đặt lại cũng như lấy ra số giờ làm việc và số ngày công. Thêm vào đó, class này cần được trang bị các hàm quá tải toán tử:

- Toán tử +: Khi hai đối tượng NumDays được cộng lại với nhau, thì toán tử + sẽ được quá tải để trả về tổng số giờ của 2 đối tượng.

- Toán tử –: Khi một đối tượng NumDays trừ cho một đối tượng NumDays khác, thì toán tử - sẽ được quá tải để trả về số giờ chênh lệch giữa số giờ của 2 đối tượng.
- Toán tử tăng lên ++ tiền tố và hậu tố: Các toán tử này sẽ làm tăng số giờ được lưu trữ trong đối tượng. Khi số giờ tăng thì số ngày phải được tự động cập nhật tăng theo.
- Toán tử giảm xuống -- tiền tố và hậu tố: Các toán tử này sẽ làm giảm đi số giờ đang được lưu trữ trong đối tượng. Khi số giờ được giảm, số ngày phải được tự động cập nhật theo.

Bài 5: Class TimeOff

Ghi chú: Bài này giả định bạn đã hoàn thành xong Bài 4.

Thiết kế một class có tên là TimeOff. Mục đích của class này là để theo dõi một nhân viên nghỉ ốm, nghỉ phép, nghỉ không hưởng lương. Class sẽ có các members là các đối tượng thuộc với NumDays đã được mô tả trong Bài 4, cụ thể:

- *maxSickDays*: Là một đối tượng NumDays ghi lại số lượng ngày tối đa của một nhân viên có thể nghỉ ốm.
- *sickTaken*: Là một đối tượng NumDays ghi lại số lượng ngày nghỉ ốm mà nhân viên đã sử dụng.
- *maxVacation*: Là một đối tượng NumDays ghi lại số lượng ngày tối đa nghỉ phép mà nhân viên vẫn được hưởng lương.
- *vacTaken*: Là một đối tượng NumDays ghi lại số lượng ngày nghỉ phép mà nhân viên đã sử dụng.
- *maxUnpaid*: Là một đối tượng NumDays ghi lại số lượng ngày nghỉ tối đa mà nhân viên không được hưởng lương.
- *unpaidTaken*: Là một đối tượng NumDays ghi lại số lượng ngày nghỉ không được trả lương mà nhân viên đã sử dụng.

Thêm vào đó, class TimeOff cần có các member làm nhiệm vụ lưu giữ thông tin Tên nhân viên, Mã định danh nhân viên. Nó có cấu tử và các member function tương ứng để ghi vào hoặc lấy ra thông tin trong mỗi một đối tượng TimeOff.

Xác nhận Input: Chính sách của công ty quy định rằng một nhân viên không được nghỉ nhiều hơn 240 giờ nghỉ có lương. Do đó, class không nên cho phép đối tượng maxVacation lưu trữ một giá trị lớn hơn số này.

Bài 6. Báo cáo nhân sự

Ghi chú: Bài này giả định bạn đã làm xong Bài 4 và Bài 5.

Viết một chương trình sử dụng sử dụng một thể hiện của class TimeOff được thiết kế trong Bài 5. Chương trình sẽ yêu cầu người dùng nhập vào số tháng mà nhân viên này đã làm việc cho công ty. Sau đó, nó sẽ sử dụng đối tượng TimeOff để tính toán và hiển thị số lượng ngày nghỉ có phép tối đa và ngày nghỉ ốm tối đa của nhân viên. Nhân viên được hưởng 12 giờ nghỉ phép và 8 giờ nghỉ ốm mỗi tháng.

Bài 7. Class Month

Thiết kế một lớp có tên là Month. Lớp này có các private members như sau:

- *name*: Một chuỗi chứa tên của một tháng, như là “January”, “February”,...
- *monthNumber*: Một biến số nguyên chứa số đại diện cho tháng. Ví dụ, January là số 1, February là số 2, v.v.. Các giá trị hợp lệ cho biến này là từ 1 đến 12.

Thêm vào đó, cung cấp các member function sau:

- Một cấu tử mặc định để đặt giá trị monthNumber bằng 1 và name bằng “January”.
- Một cấu tử nhận vào tên của tháng như một đối số. Nó sẽ đặt cho biến name giá trị được truyền vào và đặt cho biến monthNumber thành giá trị đúng.
- Một cấu tử nhận vào một số của tháng như một đối số. Nó sẽ đặt cho biến monthNumber giá trị được truyền vào và đặt cho biến name thành tên tháng đúng.
- Các hàm set và get thích hợp cho các biến name và monthNumber.
- Các hàm quá tải toán tử ++ tiền tố và hậu tố để tăng giá trị của biến monthNumber và đặt biến name thành tên tháng tiếp theo. Nếu monthNumber đang là 12 thì khi hàm này được thực thi, monthNumber sẽ được đặt là 1, và biến name sẽ được đặt là “January”.
- Các hàm quá tải toán tử -- tiền tố và hậu tố để giảm giá trị của biến monthNumber và đặt biến name thành tên tháng trước đó. Nếu monthNumber đang là 1 thì khi hàm này được thực thi, monthNumber sẽ được đặt là 12 và biến name sẽ được đặt là “December”.

Bạn cũng quá tài các toán tử << của cout và toán tử >> của cin để thực hiện với class Month. Sau đó hãy chạy demo class này trong một chương trình.

Bài 8. Chỉnh sửa class Date

Chỉnh sửa class Date trong Bài 1 của Chương 13. Phiên bản mới này sẽ có các toán tử đuược quá tải như sau:

- Toán tử tăng lên ++ tiền tố và hậu tố: Các toán tử này làm tăng giá trị ngày của đối tượng đó lên 1 đơn vị.
- Toán tử giảm xuống -- tiền tố và hậu tố: Các toán tử này làm giảm giá trị ngày của đối tượng đó đi 1 đơn vị.
- Toán tử trừ –: Nếu một đối tượng Date bị trừ đi một đối tượng Date khác, thì toán tử sẽ trả về số lượng ngày cách biệt nhau giữa 2 ngày đó. Ví dụ: 18/6/2021 – 10/6/2021 thì kết quả trả về sẽ là 8. Đồng thời, nếu một đối tượng Date trừ đi một số n ngày nào đó, thì sẽ trả về ngày cách ngày đó n ngày. Ví dụ: 18/6/2021 – 8 thì kết quả sẽ là 10/6/2021.
- Toán tử xuất << của cout: Toán tử này sẽ in ra ngày theo đúng định dạng: "April 18, 2014".
- Toán tử nhập >> của cin: Toán tử này sẽ nhắc người dùng nhập vào một ngày để lưu được vào trong đối tượng Date

***Chú ý:** Class Date sẽ hoạt động theo các nguyên tắc sau:

- Khi một ngày được đặt cho giá trị là ngày cuối cùng của tháng, thì ngày tiếp theo sẽ là ngày đầu tiên của tháng sau.
- Khi một ngày được đặt cho giá trị là ngày cuối cùng của năm, thì ngày tiếp theo sẽ là ngày đầu tiên của năm sau.
- Khi một ngày được đặt cho giá trị là ngày đầu tiên của tháng, nếu bị trừ đi thì sẽ trở thành ngày cuối cùng của tháng trước đó.
- Khi một ngày được đặt cho giá trị là ngày 1 tháng 1 và bị trừ đi, nó sẽ trở về ngày 31/12 của năm trước đó.

Yêu cầu xác nhận Input: Toán tử >> sẽ không cho phép nhập ngày không hợp lệ. Ví dụ: 39/13/2021 là một ngày không được phép.

Bài 9. Class FeetInches (Bắt buộc)

Thiết kế một class có tên là FeetInches, có nhiệm vụ lưu trữ một độ đo khoảng cách theo đơn vị feet và inches. Thông tin về số feet và inches của một vật thể là hai số nguyên, được quy về định dạng chuẩn sau khi một đối tượng FeetInches được tạo. Ví dụ:

3 feet 14 inches sẽ được điều chỉnh lại theo chuẩn là 4 feet 2 inches.

5 feet -2 inches sẽ được điều chỉnh lại theo chuẩn là 4 feet 10 inches.

(Quy ước 1 feet = 12 inches)

Code tham khảo: <<*xem ở trang sau*>>

Yêu cầu:

- Thêm vào class FeetInches các hàm quá tải toán tử so sánh: <=, >=, và toàn từ !=
- Demo khả năng của class trong một chương trình đơn giản.

Nội dung file FeetInches.h

```
#ifndef FEETINCHES_H
#define FEETINCHES_H
class FeetInches
{
    private:
        int feet;
        int inches;
        void simplify(); // định nghĩa trong file FeetInches.cpp
    public:
        // Cấu tử
        FeetInches(int f = 0, int i = 0)
        {
            feet = f;
            inches = i;
            simplify();
        }
        // các hàm set
        void setFeet(int f)
        {
            feet = f;
        }
        void setInches(int i)
        {
            inches = i;
            simplify();
        }
        // các hàm get
        int getFeet() const
        {
            return feet;
        }
        int getInches() const
        {
            return inches;
        }
        // các hàm quá tải toán tử
        FeetInches operator + (const FeetInches &); // quá tải toán tử +
        FeetInches operator - (const FeetInches &); // quá tải toán tử -
}
#endif
```

Nội dung file FeetInches.cpp

```cpp
#include<cstdlib>  // sử dụng hàm tính giá trị tuyệt đối abs
#include "FeetInches.h"

void FeetInches::simplify()
{
    if(inches >=12)
    {
        feet += (inches/12)
        inches = inches%12;
    }
    else if (inches < 0)
    {
        feet -= ((abs(inches)/12) + 1);
        inches = 12 - (abs(inches) % 12);
    }
}

//*********************************************************
// quá tại toán tử + (hai ngôi)                           *
//*********************************************************
FeetInches FeetInches::operator + (const FeetInches $right)
{
    FeetInches temp;
    temp.inches = inches + right.inches;
    temp.feet = feet + right.feet;
    temp.simplify();
    return temp;
}

//*********************************************************
// quá tại toán tử - (hai ngôi)                           *
//*********************************************************
FeetInches FeetInches::operator - (const FeetInches $right)
{
    FeetInches temp;
    temp.inches = inches - right.inches;
    temp.feet = feet - right.feet;
    temp.simplify();
    return temp;
}
```

Bài 10: Doanh số bán hàng của công ty (Bắt buộc)

Một công ty có sáu bộ phận, mỗi bộ phận chịu trách nhiệm bán hàng cho các vị trí địa lý khác nhau. Thiết kế một lớp DivSales lưu dữ liệu bán hàng cho một bộ phận, với các member sau đây:

- Một mảng với bốn thành phần lưu giữ số liệu bán hàng cho bốn quý của mỗi bộ phận.
- Một biến static trong phạm vi private để chứa tổng doanh số bán hàng cho tất cả các bộ phận trong vòng một năm.
- Một member function nhận vào bốn đối số, trong đó mỗi đối số được giả định là doanh số bán hàng trong một quý. Giá trị của các đối số sẽ được copy vào trong mảng lưu trữ dữ liệu bán hàng.Tổng của bốn đối số này sẽ được cộng dồn vào biến static để lưu giữ doanh số bán hàng của công ty theo năm.
- Một function nhận vào một đối số là số nguyên nằm trong khoảng 0-3. Đối số này sẽ được sử dụng để làm một chỉ số để truy xuất dữ liệu bán hàng cho bốn quý của các bộ phận trong công ty. Function này sẽ trả về một giá trị là một phần tử trong mảng với chỉ số được truyền vào.

Viết một chương trình tạo ra một mảng 6 đối tượng DivSales. Chương trình yêu cầu người dùng nhập vào doanh số bán hàng trong bốn quý của mỗi bộ phận trong công ty. Sau khi dữ liệu được nhập vào, chương trình sẽ hiển thị một bảng thể hiện doanh số bán hàng của các bộ phận này theo các quý. Sau đó, chương trình sẽ hiển thị tổng doanh số bán hàng của công ty trong một năm.

Yêu cầu xác nhận Input: Chỉ chấp nhận giá trị dương cho các dữ liệu bán hàng nhập cho từng quý.

Bài 11: Copy constructor của class FeetInches và function multiply (Bắt buộc)

- Tạo một cấu tử sao chép (copy constructor) của class FeetInches. Cấu tử này sẽ chấp nhận đối số đầu vào là một đối tượng FeetInches bất kỳ. Nó sẽ gán thuộc tính feet bằng giá trị của thuộc tính feet trong đối tượng đối số, gán giá trị thuộc tính inches bằng giá trị của thuộc tính inches trong đối tượng đối số, tương ứng. Kết quả trả về là một đối tượng mới được copy từ đối tượng truyền vào.
- Tiếp theo, thêm vào class FeetInches một member function có tên là multiply. Function này nhận một đối tượng FeetInches là đối số. Các thuộc tính feet và

inches của đối tượng đối số sẽ được nhân lên với các thuộc tính feet và inches của đối tượng gọi hàm multiply, và một đối tượng FeetInches chứa kết quả sẽ được trả về từ hàm multiply này.

Bài 12: Class LandTract

Tạo một class có tên LandTract bao gồm hai đối tượng FeetInches, một cho chiều dài của khoảnh đất và một cho chiều rộng. Class phải có một member function trả về diện tích của khoảnh đất. Demo phần cài đặt class này trong một chương trình có yêu cầu người dùng nhập vào chiều dài và chiều rộng của hai khoảnh đất. Chương trình sẽ hiển thị diện tích của mỗi vùng đất và cho biết các vùng đó có kích thước bằng nhau hay không.

Bài 13: Carpet Calculator

Công ty Westfield Carpet đã yêu cầu bạn viết một ứng dụng để tính giá của những tấm thảm trải sàn (carpet) cho một căn phòng hình chữ nhật. Để tính được giá, bạn phản nhân diện tích (chiều dài x chiều rộng) với giá mỗi mét foot vuông của thảm. Ví dụ, diện dích mặt sàn có chiều dài 12 feet và chiều rộng 10 feet sẽ là 120 feet vuông. Để trải đủ thảm trên mặt sàn đó, với giá thành 8 đô la mỗi foot vuông thì sẽ tốn 120 x 8 = 960 đô la.

Đầu tiên, bạn nên tạo một class có tên là RoomDimension có hai đối tượng FeetInches là thuộc tính: một cho chiều dài của căn phòng và một cho chiều rộng. (Bạn nên sử dụng phiên bản của class FeetInches mà bạn đã tạo trong Bài 11 với việc bổ sung các member function cần thiết. Bạn có thể sử dụng function này để tính diện tích của căn phòng.) Class RoomDimension phải có một member function trả về diện tích của căn phòng thông qua một đối tượng FeetInches.

Tiếp theo, bạn nên tạo một lớp có tên là RoomCarpet có đối tượng RoomDimension làm thuộc tính. Class này cũng phải có một thuộc tính chi phí của tấm thảm trên mỗi foot vuông. Class RoomCarpet phải có member function trả về tổng chi phí của thảm.

Khi bạn đã hoàn thành việc viết các lớp này, hãy sử dụng chúng trong một ứng dụng để yêu cầu người dùng nhập vào kích thước các chiều của một căn phòng và giá mỗi foot vuông của tấm thảm họ muốn mua. Sau đó ứng dụng sẽ hiển thị tổng chi phí của thảm.

Bài 14: Chương trình mô phỏng vé đỗ xe

Bạn cần thiết kế một bộ các class để chúng hoạt động cùng nhau trong việc giả lập một văn phòng cảnh sát chuyên quản lý vé đỗ xe. Các class bạn cần xây dựng là:

- Class ParkedCar: class này nên mô phỏng các xe đã đỗ. Nhiệm vụ của class này là:
  + Để lưu giữ thông tin hãng xe, model, màu sắc, mã số giấy phép, và thời gian xe đã đỗ tính bằng đơn vị phút.
- - Class ParkingMeter: class này nên mô phỏng đồng hồ tính giờ đỗ xe. Nhiệm vụ của class này là:

  + Để lưu giữ thông tin thời gian đỗ xe đã được chủ xe mua.

- Class ParkingTicket: class này mô phỏng vé đỗ xe. Class này có nhiệm vụ:
  + Để báo cáo các thông tin hãng xe, model, màu sắc, và mã số giấy phép của chiếc xe đã đỗ trái phép.
  + Để báo cáo số tiền phạt, với 25 đô la cho giờ đầu tiên (hoặc dưới một giờ đầu tiên) của chiếc xe đã đỗ trái phép, cộng thêm 10 đô la cho mỗi giờ tiếp theo (hoặc dưới 1 giờ tiếp theo) của chiếc xe đã đỗ trái phép.
  + Để báo cáo tên và số hiệu của cảnh sát xuất vé.
- Class PoliceOfficer: class này mô phỏng một cảnh sát đang kiểm tra những chiếc xe ô tô đang đỗ. Nhiệm vụ của class này là:
  + Để lưu giữ tên cảnh sách và số hiệu của cảnh sát.
  + Để xem xét đối tượng ParkedCar, kiểm tra đối tượng ParkingMeter, và xác định xem thời gian của xe ô tô đó đã hết hạn đỗ hay chưa.
  + Để phạt vé đỗ xe (tạo ra một đối tượng ParkingTicket) nếu thời gian đỗ của xe đó đã hết.

Viết một chương trình demo việc hoạt động của các lớp này khi kết hợp với nhau.

Bài 15: Chương trình mô phỏng thiết bị ô tô (Bắt buộc)

Đối với bài này, bạn sẽ thiết kế một bộ các class để chúng cùng hoạt động trong việc mô phỏng đồng hồ đo nhiên liệu và đồng hồ đo đường đi của ô tô. Các lớp bạn cần xây dựng là:

- Class FuelGauge: class này sẽ mô phỏng đồng hồ đo nhiên liệu. Nó có nhiệm vụ:

    + Để biết lượng nhiên liệu còn lại của xe được đo bằng đơn vị gallon, trong thời điểm hiện tại.
    + Để báo cáo lượng nhiên liệu còn lại của xe được đo bằng đơn vị gallon, trong thời điểm hiện tại.
    + Có khả năng tăng lượng nhiên liệu hiện tại lên 1 gallon. Điều này mô phỏng việc đổ nhiên liệu vào cho xe ô tô. (Với khả năng chứa tối đa của xe ô tô là 15 gallon.)
    + Có khả năng giảm lượng nhiên liệu hiện tại đi 1 gallon, nếu lượng nhiên liệu còn lại trong xe lớn hơn 0 gallon. Điều này mô phỏng việc đốt cháy nhiên liệu khi xe ô tô chạy.
- Class Odometer: class này mô phỏng đồng hồ đo đường đi của ô tô. Class này có nhiệm vụ:
    + Để biết số dặm hiện tại của ô tô.
    + Để báo cáo số dặm hiện tại của ô tô.
    + Có khả năng tăng số dặm hiện tại lên 1 dặm. Giá trị số dặm có khả năng hiển thị trên đồng hồ tối đa là 999,999 dặm. Khi đạt được số dặm ở mức tối đa này, đồng hồ sẽ chuyển về hiện thị số dặm bằng 0.
    + Có khả năng tương tác với đối tượng FuelGauge. Nó sẽ làm giảm số nhiên liệu hiện tại của đối tượng FuelGauge đi 1 gallon mỗi khi xe đi được 24 dặm. (Xe ô tô đi 24 dặm sẽ tiêu tốn 1 gallon).

Mô phỏng các class trên bằng việc tạo ra các thể hiện của mỗi lớp. Mô phỏng việc đổ nhiên liệu cho ô tô, và sau đó chạy một vòng lặp để tăng số dặm xe chạy được cho đến khi hết sạch nhiên liệu. Trong khi mỗi vòng lặp được thực hiện, hãy in ra số dặm hiện tại của xe và lượng nhiên liệu mà nó chứa.