BÁO CÁO BÀI TẬP 3

**Môn học: Lập trình hướng đối tượng**

**Tên chủ đề: báo cáo bài tập 3**

*GVHD: Nguyễn Hữu Quyền*

*Ngày báo cáo: 18/04/2023*

**Nhóm: 02**

1. **THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: IT002.N28.2

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Nguyễn Hải Phong | 22521088 | 22521088@gm.uit.edu.vn |
| 2 | Hồ Trung Kiên | 22520704 | 22520704@gm.uit.edu.vn |

1. **NỘI DUNG THỰC HIỆN:[[1]](#footnote-1)**

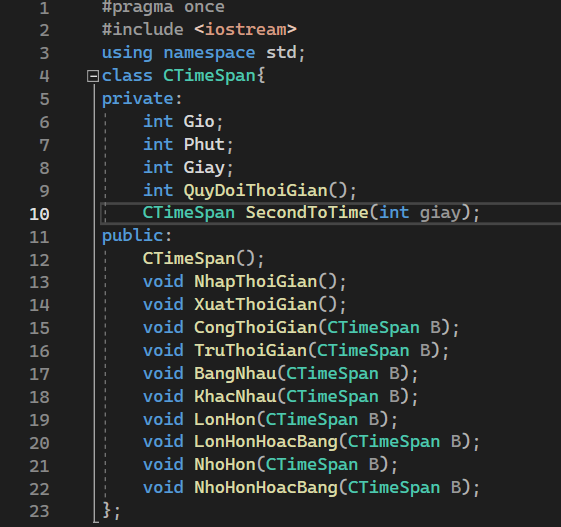
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Công việc** | **Kết quả tự đánh giá** | **Người đóng góp** |
| 1 | Câu hỏi 01 | 100% | Nguyễn Hải Phong |
| 2 | Câu hỏi 02 | 100% | Nguyễn Hải Phong |
| 3 | Câu hỏi 03 | 100% | Hồ Trung Kiên |
| 4 | Câu hỏi 04 | 100% | Nguyễn Hải Phong |

**Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.**

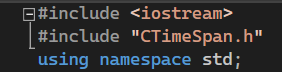
BÁO CÁO CHI TIẾT BÀI THỰC HÀNH

## Câu hỏi 1

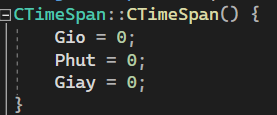
* + - Tài nguyên: Lớp CTimeSpan có ba thành phần dữ liệu là giờ, phút, giây và các hàm nhập, xuất thời gian, các hàm phép toán cần thiết.
* Mô tả/mục tiêu: Định nghĩa lớp dữ liệu CTimeSpan để biểu diễn khái niệm khoảng thời gian, các hàm thành phần và các phép toán cần thiết (+, -, ==,!=, >, >=, <=), hàm xuất, nhập.
* Các bước thực hiện/Phương pháp thực hiện:
  + - * Bước 1: Tạo ra 1 header file tên CTimeSpan.h để khai báo sơ lược những thành phần cần có của lớp dữ liệu CTimeSpan.h. Lớp dữ liệu CTimeSpan.h gồm các thành phần private là thuộc tính giờ, phút và giây (cùng kiểu dữ liệu int) cũng như phương thức quy đổi giờ sang giây (hàm QuyDoiThoiGian) và ngược lại (hàm SecondtoTime). Trong phần public sẽ là các phương thức nhập xuất thời gian, thực hiện cộng trừ và so sánh giữa 2 thời gian được nhập vào.



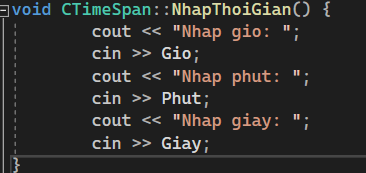
* Bước 2: Khởi tạo một chương trình tên CTimeSpan.cpp và include thư viện iostream và file header vừa được tạo ra ở trên để viết ra chức năng của những phương thức vừa được khởi tạo trong nó.



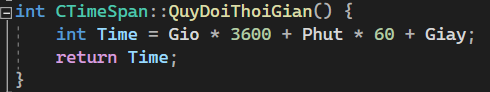
Ở hàm khởi tạo (constructor), ta sẽ cho khai báo biến của giờ, phút và giây, có thể cho mặc định cả ba đều bằng 0 (hoặc 1 giá trị khác nhưng không nên vượt quá giới hạn cho phép của int vì nếu không sẽ gây lỗi integer overflow)



Sau đó là hàm nhập hay còn gọi là hàm cập nhật. Hàm này sẽ cho phép người dùng nhập vào giá trị của thời gian theo ý họ.



Hàm dùng để chuyển đổi từ thời gian sang giây và ngược lại dùng để thuận tiện cho việc tính toán cộng trừ giữa hai thời gian. Ở hàm này, ta sẽ tiến hành chuyển từ giờ, phút, giây sang giây bằng cách quy đổi giờ và phút đều sang giây và cộng lại (1 giờ = 60 phút = 3600 giây, 1 phút = 60 giây).

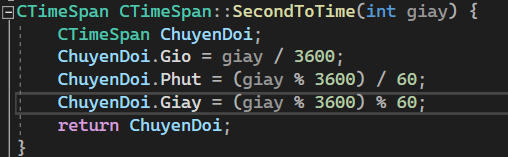


Trong khi hàm chuyển từ giây được truyền vào sang giờ, phút và giây cần truyền 1 biến là số giây và được chuyển đổi bằng cách:

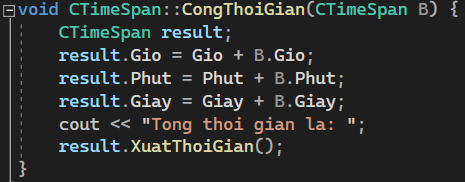
**Giờ = giây / 3600 (1 giờ = 3600 giây)**

**Phút = (giây % 3600) / 60 (1 phút = 60 giây)**

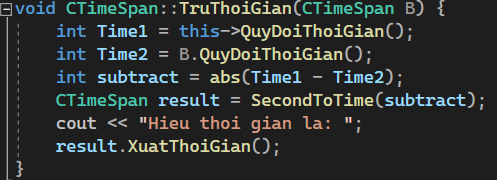
**Giây = (giây % 3600) % 60 (lấy phần dư sau khi quy đổi sang giờ và phút)**

****

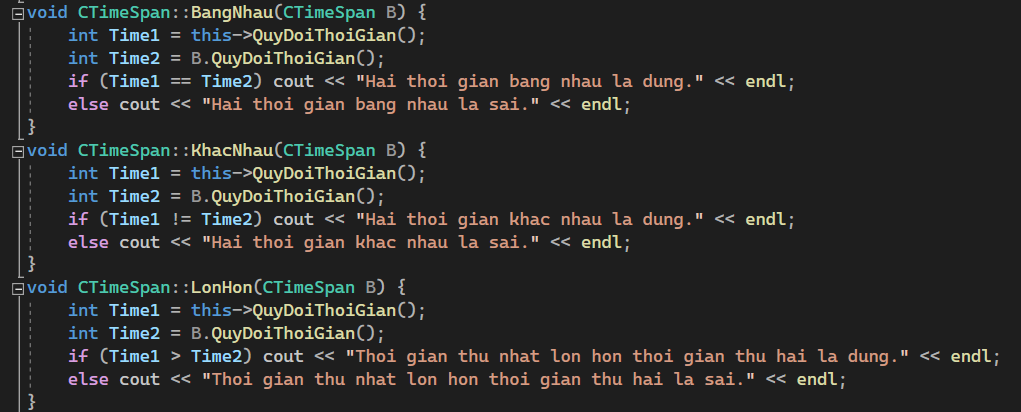
Hàm cộng 2 thời gian nhập vào cần truyền vào thêm một biến B có kiểu dữ liệu CTimeSpan để thực hiện phép cộng, được thực hiện bằng cách thực hiện phép cộng giữa giờ, phút và giây của 2 thời gian. Sau đó cuối cùng tiến hành xuất kết quả thông qua hàm xuất.

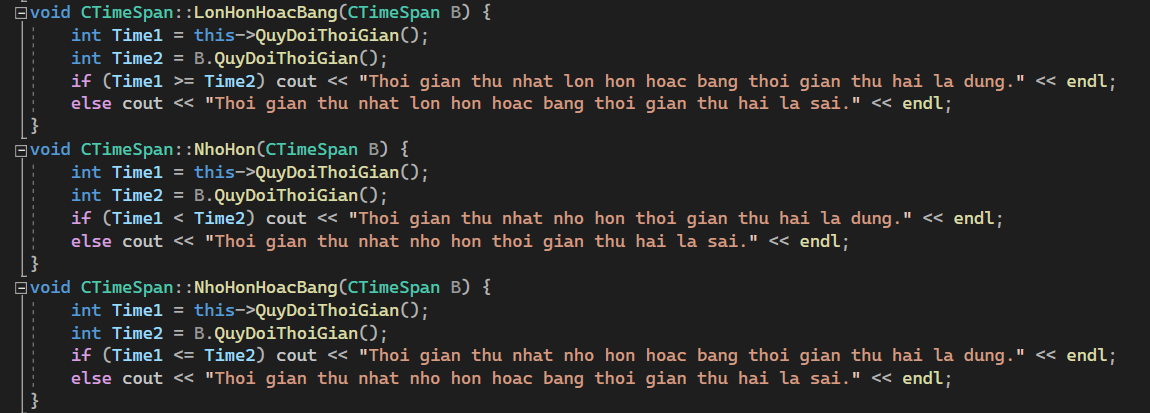


Hàm trừ 2 thời gian nhập vào cần truyền vào thêm một biến B có kiểu dữ liệu CTimeSpan để thực hiện phép trừ, được thực hiện bằng cách quy đổi 2 thời gian được nhập vào sang giây bằng hàm chuyển đổi sang giây đã được thực hiện ở trên. Sau đó ta tiến hành trừ 2 thời gian đó trong cùng đơn vị là giây. Vì không biết thời gian nào được nhập vào lớn hơn nên ta tiến hành lấy trị tuyệt đối cho phép trừ này (gọi đến hàm abs). Sau khi trừ xong, tiến hành quy đổi kết quả về thời gian (giờ, phút, giây) và gọi đến hàm xuất để xuất ra kết quả.

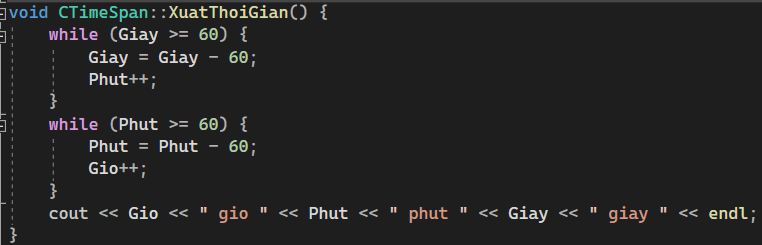


Đến hàm so sánh giữa hai thời gian thì ta cần truyền vào thêm một biến B có kiểu dữ liệu là lớp CTimeSpan và tiến hành quy đổi 2 thời gian được nhập vào sang cùng một đơn vị giây thông qua hàm chuyển đổi thời gian sang giây. Sau đó tiến hành so sánh thời gian này và xuất ra kết quả so sánh đó.

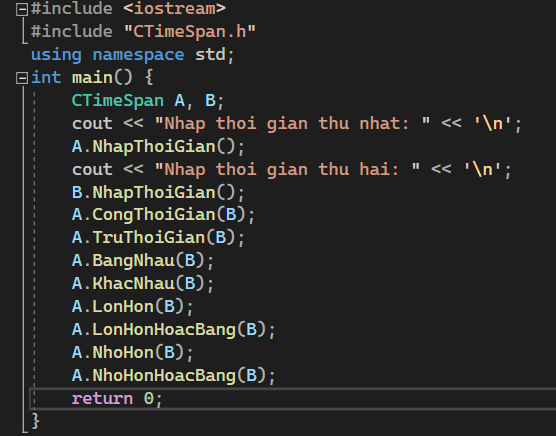




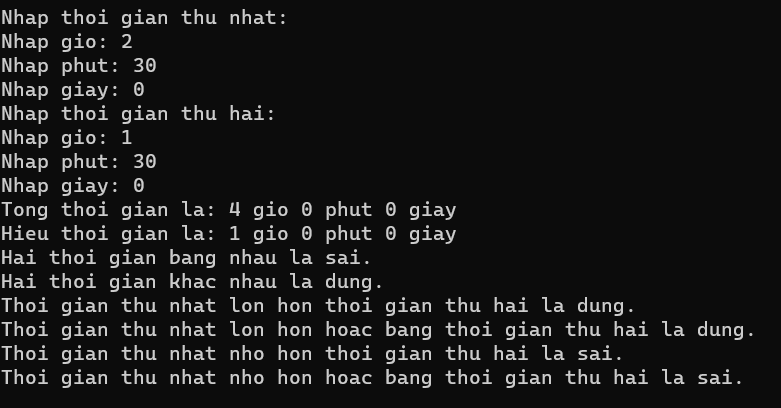
Cuối cùng là hàm xuất dùng để xuất ra giờ, phút, và giây. Trước khi xuất ra thì ta kiểm tra tính hợp lệ của giờ, phút và giây. Nếu như giây có lớn hơn hoặc bằng 60 thì ta quy đổi số giây dư sang phút, tương tự với nếu phút lớn hơn hoặc bằng 60 thì ta quy đổi số phút dư sang giờ,



* Bước 3: Tạo thêm một file C++ dùng để thực thi các phương thức tạo ra trong lớp CTimeSpan trong hàm main. Đồng thời include thêm file header CTimeSpan.h để có thể gọi đến các phương thức trong lớp CTimeSpan. Ta lần lượt khai báo 2 biến A và B có kiểu dữ liệu là lớp CTimeSpan. Và đồng thời gọi đến hàm nhập để nhập 2 thời gian, hàm cộng trừ 2 thời gian đó, cũng như hàm so sánh 2 thời gian đó.

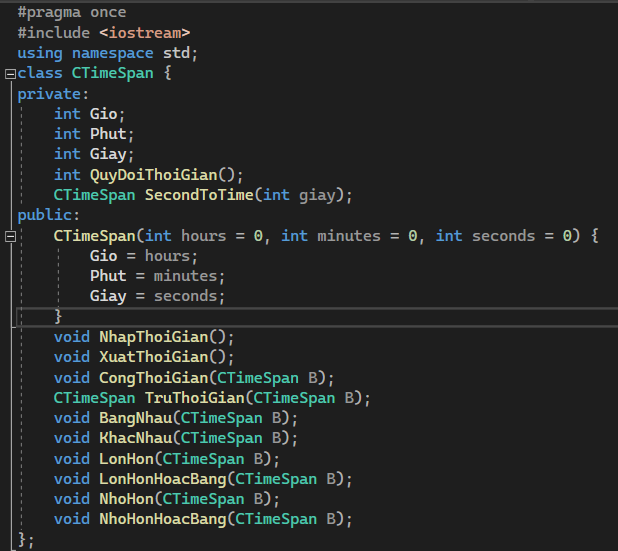


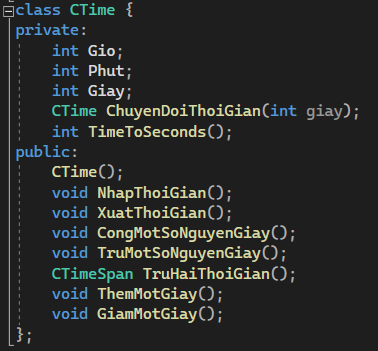
Ví dụ demo:



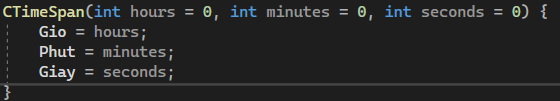
## Câu hỏi 2

* Tài nguyên: Lớp CTimeSpan với các thành phần dữ liệu là giờ, phút, giây và các hàm nhập, xuất, và các hàm phép toán cần thiết (cộng, trừ) hai khoảng thời gian và hàm so sánh, lớp CTime với các thành phần dữ liệu là giờ, phút, giây và các hàm nhập, xuất thời điểm và các phép toán cộng trừ số nguyên giây, hàm tính khoảng thời gian giữa 2 mốc thời gian và hàm thêm bớt một giây
* Mô tả/Mục tiêu: Định nghĩa lớp CTime biểu diễn khái niệm thời điểm có các thành phần giờ phút giây. Định nghĩa các phép toán +, - (cộng, trừ thêm một số nguyên giây), - (phép trừ hai CTime để được một CTimSpan), ++, -- (thêm bớt một giây), hàm xuất, nhập.
* Các bước thực hiện/Phương pháp thực hiện:
  + - * Bước 1: Tạo 1 file header C++ là CTime.h chứa các phần sơ lược của class CTime và class CTimeSpan. Trong đó, CTimeSpan chứa các thuộc tính giờ, phút, giây, hàm chuyển đổi giữa thời gian sang giây và ngược lại trong private và các hàm nhập xuất, các hàm tính toán cần thiết trong public. Còn lớp CTime cũng chứa các thuộc tính giờ, phút, giây, hàm chuyển đổi giữa thời gian sang giây và ngược lại trong private và các hàm nhập xuất mốc thời gian, hàm tính toán cộng trừ mốc thời gian cho một số nguyên giây nhất định, hàm tính khoảng thời gian giữa 2 mốc thời gian và hàm tăng giảm mốc thời gian 1 giây.

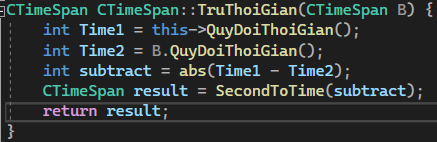




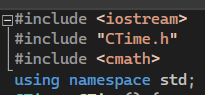
Về cơ bản trong class CTimeSpan thì hầu hết các chức năng của các phương thức đều không có gì thay đổi so với câu hỏi 1 nhưng chỉ có thay đổi ở 2 chỗ để phù hợp cho việc tính toán 2 mốc thời gian trả về 1 CTimeSpan. Đầu tiên là hàm constructor của lớp CTimeSpan ta sẽ truyền đối số vào (giờ, phút, và giây, cho mặc định bằng 0 nếu không truyền đối số vào) để khởi tạo giá trị của 1 biến CTimeSpan sẽ được sử dụng trong hàm tính khoảng thời gian giữa 2 mốc thời gian.



Còn hàm trừ 2 khoảng thời gian trong CTimeSpan thì thay vì ta xuất ra kết quả màn hình sau khi thực hiện tính toán xong thì giờ hàm này ta sẽ chuyển kiểu trả về từ void sang lớp CTimeSpan để trả về kết quả của hàm này, riêng thuật toán thì vẫn không có gì thay đổi so với hàm trừ của bài 1.



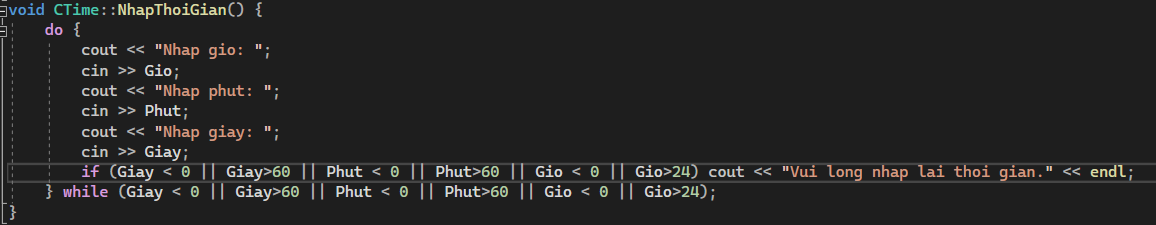
* Bước 2: Tạo ra 1 file C++ tên CTime.cpp để viết các chức năng của các phương thức vừa định nghĩa trong lớp CTime. Đồng thời gọi đến thư viện iostream và file header CTime.h vừa tạo ở trên và thư viện cmath (dùng trong việc tính toán thời gian)



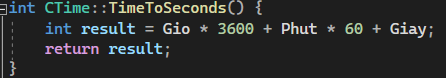
Ở hàm khởi tạo (constructor), ta sẽ cho khai báo biến của giờ, phút và giây, có thể cho mặc định cả ba đều bằng 0 (hoặc 1 giá trị khác nhưng không nên vượt quá giới hạn cho phép của int vì nếu không sẽ gây lỗi integer overflow)



Sau đó là hàm nhập hay còn gọi là hàm cập nhật. Hàm này sẽ cho phép người dùng nhập vào giá trị của mốc thời gian theo ý họ. Vì đây là mốc thời gian nên ta cần kiểm tra tính hợp lệ của mốc thời gian được nhập vào. Nếu như giờ < 0 hoặc > 24 giờ hoặc là phút, giây < 0 hoặc > 60 thì ta sẽ yêu cầu người dùng nhập lại mốc thời gian đến khi hợp lệ thì thôi.



Hàm chuyển đổi từ mốc thời gian sang giây trong phần private dùng để chuyển đổi giờ, phút và giây của mốc thời gian sang cùng 1 hệ đơn vị là giây để thuận tiện cho việc tính toán thời gian trong các hàm tính toán thời gian. Ở hàm này, ta sẽ tiến hành chuyển từ giờ, phút, giây sang giây bằng cách quy đổi giờ và phút đều sang giây và cộng lại (1 giờ = 60 phút = 3600 giây, 1 phút = 60 giây).

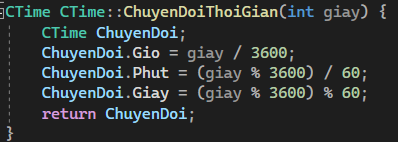


Còn hàm chuyển đổi từ giây được truyền vào trong hàm sang mốc thời gian (giờ, phút, giây) thì được thực hiện bằng cách:

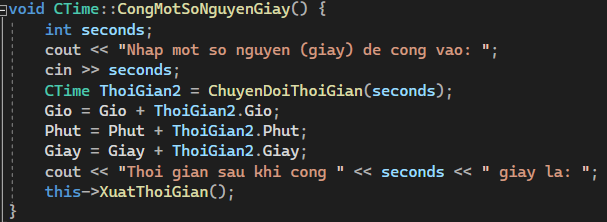
**Giờ = giây / 3600 (1 giờ = 3600 giây)**

**Phút = (giây % 3600) / 60 (1 phút = 60 giây)**

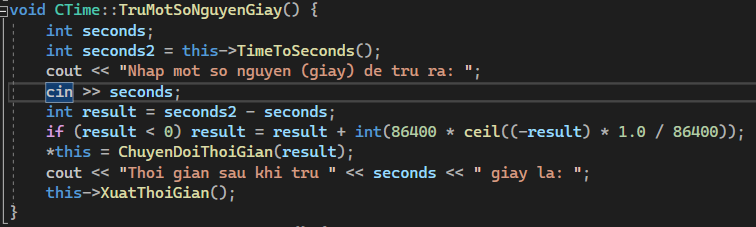
**Giây = (giây % 3600) % 60 (lấy phần dư sau khi quy đổi sang giờ và phút)**



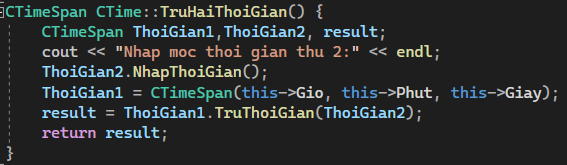
Hàm cộng mốc thời gian thêm 1 số nguyên giây sẽ cộng thêm số nguyên giây đó cho biến gọi đến hàm này. Đầu tiên ta sẽ yêu cầu người dùng nhập vào một số nguyên giây bất kỳ (giây) để thực hiện phép cộng này, tiếp đến ta sẽ gọi đến hàm chuyển đổi từ giây sang mốc thời gian và lưu vào 1 biến có kiểu dữ liệu là CTime. Sau đó ta lần lượt cộng mốc thời gian được biến gọi đến hàm với thời gian của biến vừa tạo. Sau khi thực hiện phép tính xong ta tiến hành gọi đến hàm xuất của lớp CTime để xuất ra màn hình.



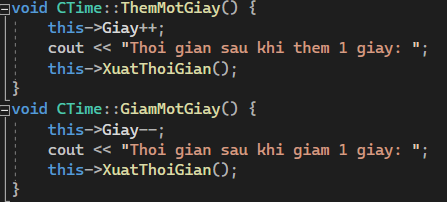
Hàm trừ mốc thời gian thêm 1 số nguyên giây sẽ cộng thêm số nguyên giây đó cho biến gọi đến hàm này. Đầu tiên ta sẽ yêu cầu người dùng nhập vào một số nguyên giây bất kỳ (giây) để thực hiện phép trừ này. Sau đó, ta sẽ gọi đến hàm chuyển đổi mốc thời gian sang giây cho mốc thời gian của biến gọi đến hàm này và lưu vào 1 biến có kiểu dữ liệu int (số nguyên). Sau đó ta thực hiện phép trừ giữa 2 số nguyên có cùng đơn vị giây và lưu vào 1 biến. Tiếp đến ta sẽ kiểm tra nếu phép trừ trên nếu nhỏ hơn 0 thì ta sẽ cộng bù thêm 1 ngày (86400 giây) để kết quả ra số dương (vì nếu phép trừ nhỏ hơn 0 thì tức là mốc thời gian sẽ quay ngược trước 1 ngày, nếu vậy thì ta phải cộng thêm 86400 giây để bù cho việc đó). Sau khi thực hiện phép tính xong ta tiến hành gọi đến hàm xuất của lớp CTime để xuất ra màn hình.



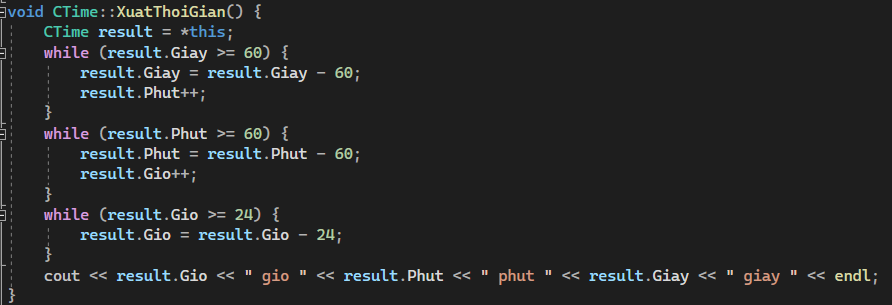
Hàm trừ 2 mốc thời gian để được một CTimeSpan thì ta phải quy định kiểu trả về của hàm là lớp CTimeSpan và đầu tiên ta sẽ yêu cầu người dùng nhập vào mốc thời gian thứ hai (cụ thể là giờ, phút, giây của mốc thời gian thứ hai) và ta sẽ lưu vào 1 biến có kiểu dữ liệu là lớp CTimeSpan. Đồng thời ta cũng phải chuyển mốc thời gian mà người dùng đã nhập vào sang cùng lớp CTimeSpan để có thể gọi đến hàm trừ 2 khoảng thời gian của lớp CTimeSpan. Để làm vậy thì ta gọi đến hàm constructor của lớp CTimeSpan ở trên và truyền vào tham số của của giờ, phút và giây của mốc thời gian gọi đến hàm trừ 2 mốc thời gian. Tiếp theo là ta sẽ gọi đến hàm trừ 2 khoảng thời gian có trong lớp CTimeSpan và lưu vào một biến có cùng kiểu dữ liệu CTimeSpan. Cuối cùng ta sẽ trả về biến lưu kết quả của phép trừ 2 CTimeSpan ấy.



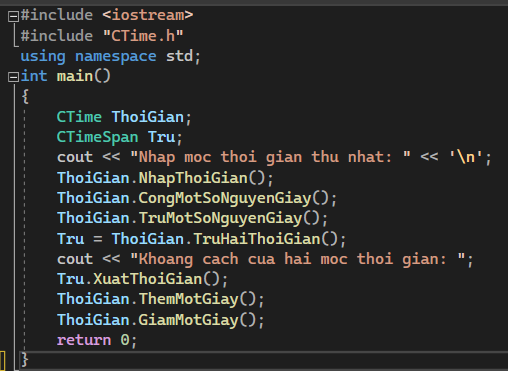
Đến hàm thêm bớt mốc thời gian 1 giây thì ta chỉ cần lấy giây của mốc thời gian gọi đến hàm này cộng trừ đi 1 giây. Sau đó ta sẽ gọi đến hàm xuất để in kết quả ra màn hình.



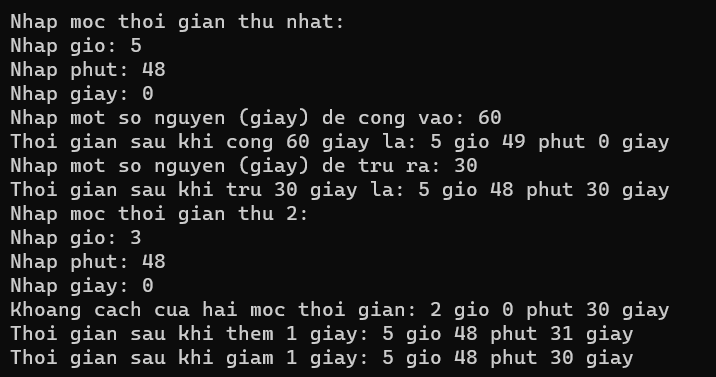
Cuối cùng là hàm xuất thì đầu tiên ta sẽ kiểm tra xem mốc thời gian gọi đến hàm xuất đã hợp lệ chưa bằng cách lần lượt kiểm tra giây, nếu giây có lớn hơn 60 thì ta trừ giây đi 60 và cộng thêm một cho phút cho đến khi giây < 60 thì dừng. Tương tự với phút thì nếu phút >=60 thì ta sẽ giảm phút đi 60 và cộng thêm 1 cho giờ đến khi phút < 60 thì dừng. Còn đối với giờ, nếu nếu như giờ >=24 thì ta sẽ trừ đi 24 đến khi nhỏ hơn 24 thì dừng. Sau đó ta sẽ xuất ra kết quả đó ra màn hình.



* Bước 3: Tạo thêm 1 file C++ chính để khởi chạy các phương thức trong các lớp. Đầu tiên ta thêm thư viện iostream và file header C++ vừa tạo để thực hiện. Đồng thời, trong hàm main thì ta tạo 1 biến có kiểu dữ liệu là lớp CTimeSpan để lưu kết quả của phép trừ 2 mốc thời gian và 1 biến có kiểu dữ liệu là lớp CTime. Sau đó ta gọi đến hàm nhập, cộng trừ 1 số nguyên giây, thêm bớt 1 giây và hàm trừ 2 mốc thời gian. Riêng hàm trừ 2 mốc thời gian thì ta sẽ lưu kết quả của hàm và biến có kiểu CTimeSpan vừa tạo và ta sẽ gọi đến hàm xuất trong lớp CTimeSpan để in kết quả trên màn hình.

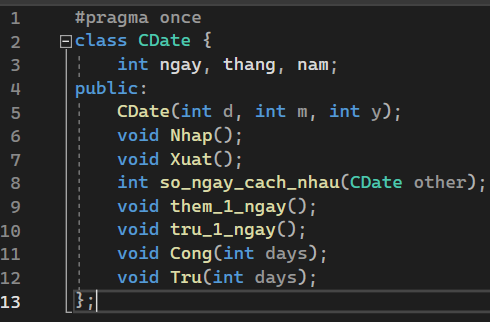


Ví dụ demo:



## Câu hỏi 3

* Tài nguyên: Định nghĩa lớp CDate biểu diễn khái niệm ngày, tháng, năm.
* Mô tả/Mục tiêu: viết các phép toán +, - (cộng, trừ thêm một số ngày), ++, -- (thêm bớt một ngày), - (khoảng cách giữa hai CDate tính bằng ngày), hàm xuất, nhập. Lưu ý viết theo dạng hàm: cong, tru, ...; không dùng overloading operator.
* Các bước thực hiện/Phương pháp thực hiện:
* Bước 1: Tạo một header file tên CDate.h dùng để khai báo sơ lược những thành phần cần có của lớp CDate. Lớp CDate chứa các thành phần private là ngay, thang và nam để biểu diễn ngày, tháng, năm. Còn trong public thì chứa các phương thức nhập, xuất đa thức và các phép cộng, trừ một ngày, cộng, trừ n ngày, và xuất tổng ngày, số ngày cách nhau giữa 2 ngày (2 object lớp CDate) với kiểu trả về int.



* Bước 2: Tạo thêm 1 file C++ tên CDate.cpp và thêm vào thư viện iostream (dùng cho hàm nhập xuất) và file header “CDate.h” vừa tạo ở trên.

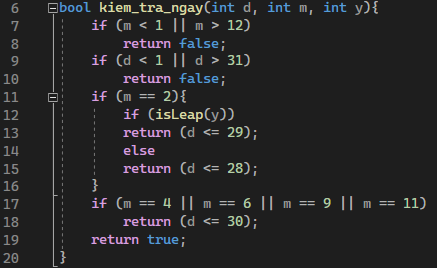
Text

Description automatically generated

bool isLeap(int year): Hàm này sẽ hổ trợ kiểm tra xem một năm có phải là năm nhuận hay không. Nếu năm đó chia hết cho 4 nhưng không chia hết cho 100 hoặc chia hết cho 400 thì nó là năm nhuận (vì ta đang sử dụng lịch Gregorian) và trả về giá trị true, ngược lại trả về giá trị false.



bool kiem\_tra\_ngay(int d, int m, int y): Hàm này kiểm tra xem một ngày có hợp lệ hay không. Nếu tháng mà ngày đó nằm trong khoảng 1 đến 12 và ngày nằm trong khoảng từ 1 đến 31 (hoặc 28 hoặc 29 hoặc 30 tùy vào tháng và năm) thì ngày đó là hợp lệ và trả về giá trị true, ngược lại trả về giá trị false.



CDate::CDate(int d,int m,int y): Đây là hàm khởi tạo của lớp CDate, được sử dụng để tạo đối tượng ngày. Hàm này truyền vào 3 tham số là ngày, tháng và năm. Nếu ngày tháng năm không hợp lệ thì yêu cầu người dùng nhập lại.

Text

Description automatically generated

void CDate::Nhap(): Hàm này được sử dụng để yêu cầu người dùng nhập ngày tháng năm cho đối tượng ngày có chức năng như một hàm cập nhật.

Text

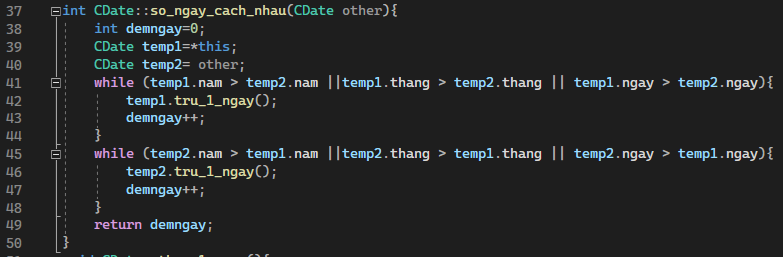
Description automatically generated

void CDate::Xuat(): Hàm này được sử dụng để xuất ngày tháng năm của đối tượng ngày theo định dạng dd/mm/yyyy.

Text

Description automatically generated

int CDate::so\_ngay\_cach\_nhau(CDate other): Hàm này tính số ngày chênh lệch giữa đối tượng ngày hiện tại và một đối tượng ngày khác (other). Để làm được việc này, hàm sử dụng hai đối tượng ngày tạm thời temp1 và temp2 và thực hiện việc trừ đi ngày cho đến khi temp1 và temp2 trùng nhau. Mỗi lần trừ đi một ngày thì biến demngay được tăng lên một đơn vị.



void CDate::them\_1\_ngay(): Hàm này được sử dụng để thêm một ngày cho đối tượng ngày hiện tại. Nếu ngày sau khi thêm không hợp lệ thì sẽ tăng tháng hoặc tăng năm.

Text

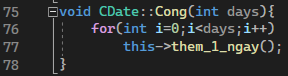
Description automatically generated

void CDate::tru\_1\_ngay(): Hàm này được sử dụng để trừ đi một ngày cho đối tượng ngày hiện tại. Nếu ngày trước khi trừ không hợp lệ

Text

Description automatically generated

**void CDate::Cong(): được sử dụng để cộng thêm một số ngày vào ngày hiện tại được đại diện bởi đối tượng CDate. Phương thức này thực hiện điều này bằng cách gọi phương thức them\_1\_ngay() days lần.**

****

**void CDate::Tru(): được sử dụng để trừ đi một số ngày khỏi ngày hiện tại được đại diện bởi đối tượng CDate. Phương thức này thực hiện điều này bằng cách gọi phương thức tru\_1\_ngay() days lần.**

**Text

Description automatically generated**

* Bước 3: Tạo thêm 1 file C++ và kèm theo các thư viện cần thiết như iostream, và file “CDate.h” cùng “CDate.cpp”.

**Text

Description automatically generated**

Trong hàm main ta thực hiện các công việc sau:

**1. Khai báo và khởi tạo các biến ngày, tháng, năm cho ngày `a`.**

**2. Khởi tạo đối tượng `a` từ lớp `CDate` với các giá trị ngày, tháng, năm vừa khởi tạo ở bước 1.**

**3. Khởi tạo đối tượng `b` từ lớp `CDate` với ngày 1/1/1.**

**4. Nhập ngày tháng năm cho ngày `b` từ bàn phím.**

**5. Tính số ngày cách nhau giữa ngày `a` và ngày `b` bằng cách gọi phương thức `so\_ngay\_cach\_nhau` của đối tượng `a` và in kết quả ra màn hình.**

**6. Tăng ngày của `a` và `b` lên 1 bằng cách gọi phương thức `them\_1\_ngay` của từng đối tượng và in ngày mới ra màn hình.**

**7. Giảm ngày của `a` và `b` đi 1 bằng cách gọi phương thức `tru\_1\_ngay` của từng đối tượng và in ngày mới ra màn hình.**

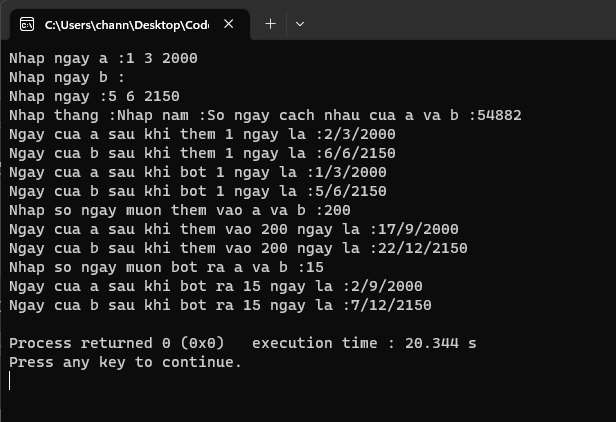
**8. Nhập số ngày cần thêm vào `a` và `b` từ bàn phím, thực hiện việc thêm ngày bằng cách gọi phương thức `Cong` của từng đối tượng và in ngày mới ra màn hình.**

**9. Nhập số ngày cần bớt khỏi `a` và `b` từ bàn phím, thực hiện việc bớt ngày bằng cách gọi phương thức `Tru` của từng đối tượng và in ngày mới ra màn hình.**

**Text

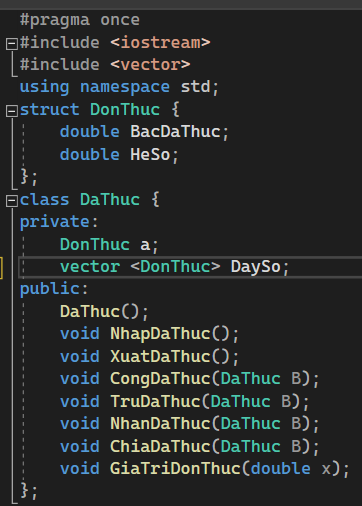
Description automatically generated**

Ví dụ demo:

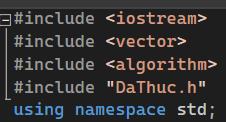


## Câu hỏi 4

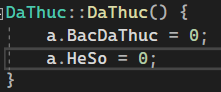
* Tài nguyên: Struct DonThuc với 2 thành phần dữ liệu là số bậc và hệ số cùng với class DaThuc với 2 thành phần dữ liệu là DonThuc với mảng để chứa các hệ số và số bậc của đa thức, các hàm nhập, xuất đa thức và các hàm phép toán cần thiết (cộng, trừ, nhân, chia)
* Mô tả/Mục tiêu: Định nghĩa lớp biểu diễn khái niệm đa thức có bậc bất kỳ với các hàm thành phần và phép toán cần thiết
* Các bước thực hiện/Phương pháp thực hiện:
* Bước 1: Tạo một header file tên DaThuc.h dùng để khai báo sơ lược những thành phần cần có của struct DonThuc cũng như các thành phần cần thiết của lớp DaThuc, trong đó struct DonThuc gồm các thành phần là hệ số cũng như bậc của đơn thức. Còn lớp đa thức thì chứa thành phần private là một đơn thức với một mảng có các phần tử là struct DonThuc (vì đa thức là tập hợp của nhiều đơn thức cộng lại). Còn trong public thì chứa các phương thức nhập, xuất đa thức và các phép toán cộng, trừ, nhân, chia 2 đa thức, và thế giá trị của x để tính giá trị đa thức, trong đó các phép toán đều cần truyền vào giá trị của biến có kiểu dữ liệu là lớp DaThuc. Ngoài ra, ta còn gọi đến thư viện iostream (dùng để nhập xuất) và thư việc vector (mảng động chứa các phần từ đơn thức)



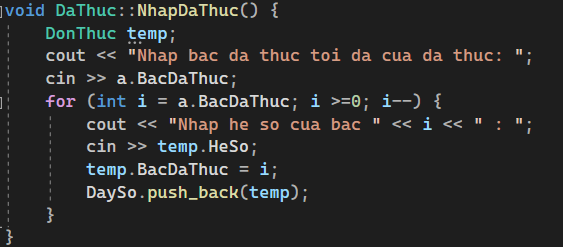
* Bước 2: Tạo thêm 1 file C++ tên DaThuc.cpp và thêm vào thư viện iostream (dùng cho hàm nhập xuất), thư viện vector (dùng để quản lý mảng gồm các đơn thức) và thư viện algorithm (dùng để phục vụ cho các hàm cộng trừ nhân chia đa thức) và file header vừa tạo ở trên.



Ở hàm khởi tạo (constructor), ta sẽ cho khai báo biến của hệ số và số bậc của đơn thức có thể cho mặc định đều bằng 0 (hoặc 1 giá trị khác nhưng không nên vượt quá giới hạn cho phép của int vì nếu không sẽ gây lỗi integer overflow)



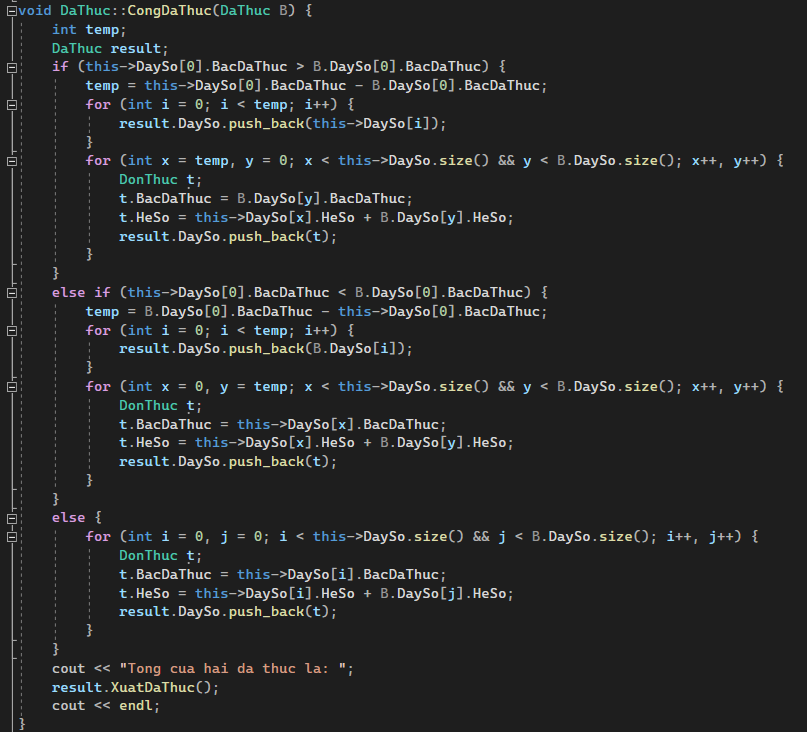
Sau đó là hàm nhập hay còn gọi là hàm cập nhật. Hàm này sẽ cho phép người dùng nhập vào giá trị của đa thức theo ý họ. Đầu tiên ta sẽ yêu cầu người dùng nhập số bậc tối đa của đa thức rồi tiếp đến là nhập hệ số của đa thức có số bậc giảm dần về 0.



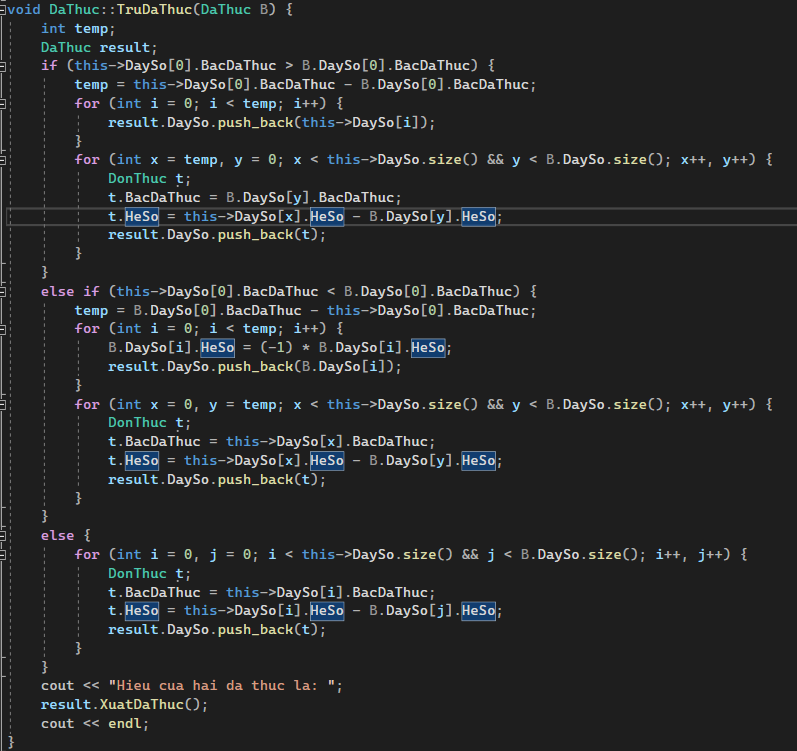
Hàm cộng hai đa thức sẽ cần truyền vào thêm một biến có kiểu dữ liệu là lớp DaThuc. Để thực hiện phép cộng thì trước hết cần kiểm tra bậc cao nhất của các hai đa thức. Nếu như số bậc cao nhất của cả hai đa thức đều bằng nhau thì ta tiến hành cộng hệ số của từng phần tử có cùng số bậc trong cả hai mảng với nhau bằng cách chỉ cộng hệ số và giữ nguyên số bậc, đồng thời truyền vào trong một mảng vector khác. Còn nếu bậc cao nhất của 2 đa thức khác nhau thì ta truyền vào mảng mới hệ số và bậc của đa thức cao hơn cho đến khi bằng với bậc cao nhất của đa thức còn lại rồi ta tính hai cộng hai đa thức có cùng bậc trong cả hai mảng với nhau bẳng cách cộng hai hệ số lại và cuối cùng truyền vào mảng khác. Ví dụ:

Nếu đa thức 1 có số bậc cao nhất là 5 và đa thức 2 có số bậc cao nhất là 3 thì vì đa thức 1 có số bậc cao hơn bậc 2 nên đầu tiên ta truyền vào mảng mới hệ số và số bậc 5 và 4 của đa thức 1 vào mảng rồi ta tiến hành cộng hai đa thức từ số bậc 3 trờ đi của cả hai đa thức.

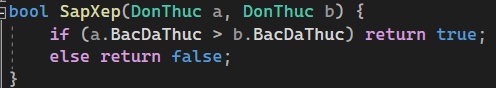
Cuối cùng ta gọi đến hàm xuất vởi mảng mới là kết quả của phép cộng của hai đa thức được nhập vào.



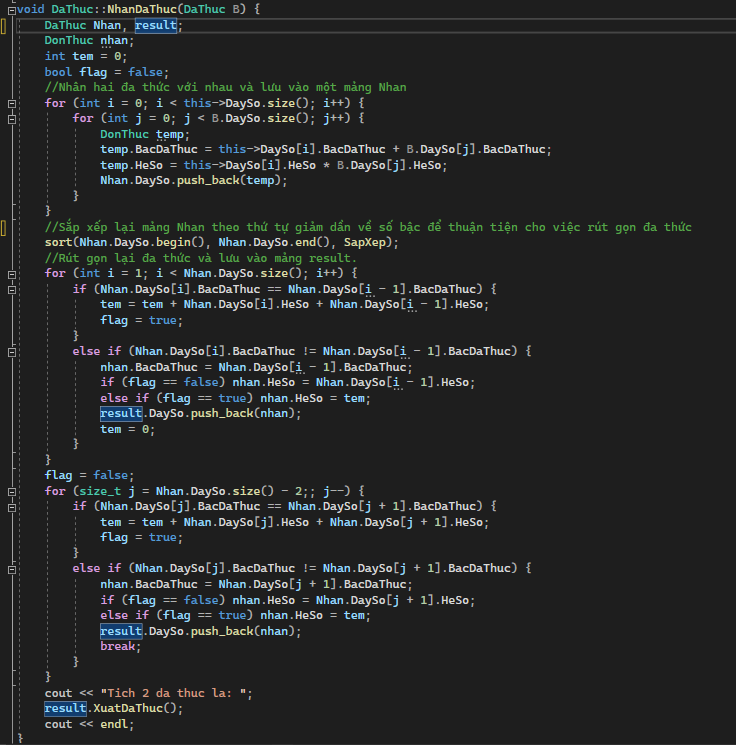
Hàm trừ hai đa thức cũng cần truyền vào một biến có kiểu dữ liệu là lớp DaThuc. Nếu như hai đa thức đều có số bậc cao nhất bằng nhau thì ta tiến hành trừ hai đa thức bằng cách trừ hệ số của hai đa thức và giữ nguyên số bậc, đồng thời thực hiện phép trừ theo chiều giảm dần về số bậc và truyền vào trong một mảng mới. Còn nếu như đa thức 1 có số bậc cao nhất cao hơn đa thức 2 thì ta tiến hành truyền các hệ số và số bậc lớn hơn của đa thức 1 vào trước đến khi bằng với số bậc cao nhất của đa thức 2 thì thực hiện phép trừ hai đa thức. Còn nếu ngược lại thì ta truyền vào mảng mới số âm của hệ số đa thức 2 và số bậc cao hơn cho đến khi bằng với số bậc cao nhất của đa thức 1 thì thực hiện phép trừ hai đa thức. Cuối cùng ta gọi đến hàm xuất cho mảng mới chứa kết quả của phép trừ hai đa thức.



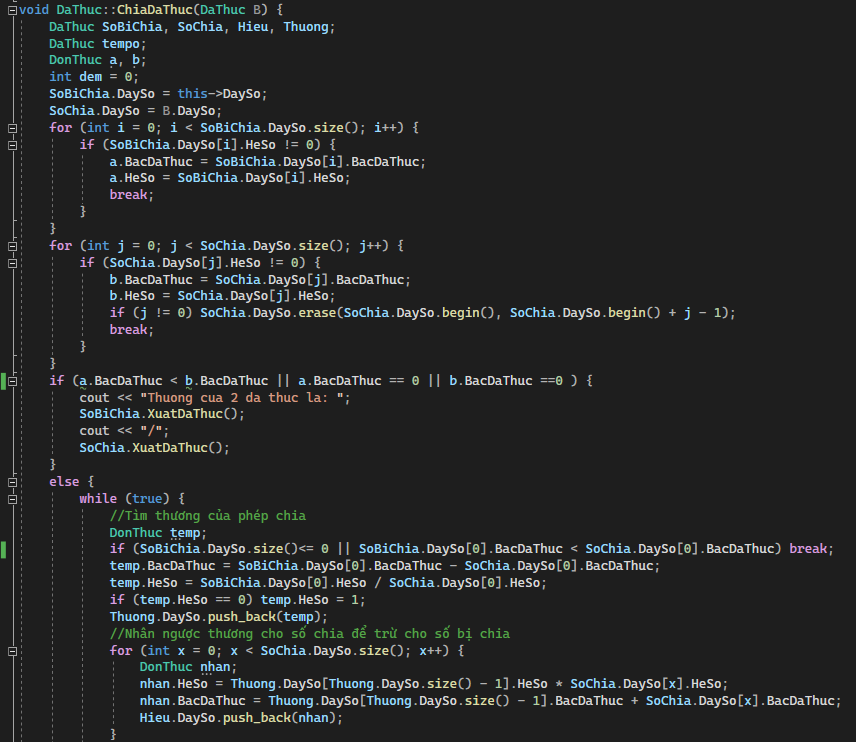
Hàm SapXep dùng để định nghĩa lại hàm sort trong thư viện algorithm, dùng để bổ trợ cho việc sắp xếp lại mảng có kiểu dữ liệu là struct DonThuc theo chiều giảm dần số bậc đến số bậc bằng 0. Ta tiến hành truyền vào 2 biến a và b có cùng kiểu struct DonThuc và hàm có kiểu trả về là bool. Nếu như số bậc của a lớn hơn thì ta trả về là true còn không thì trả về lại false.

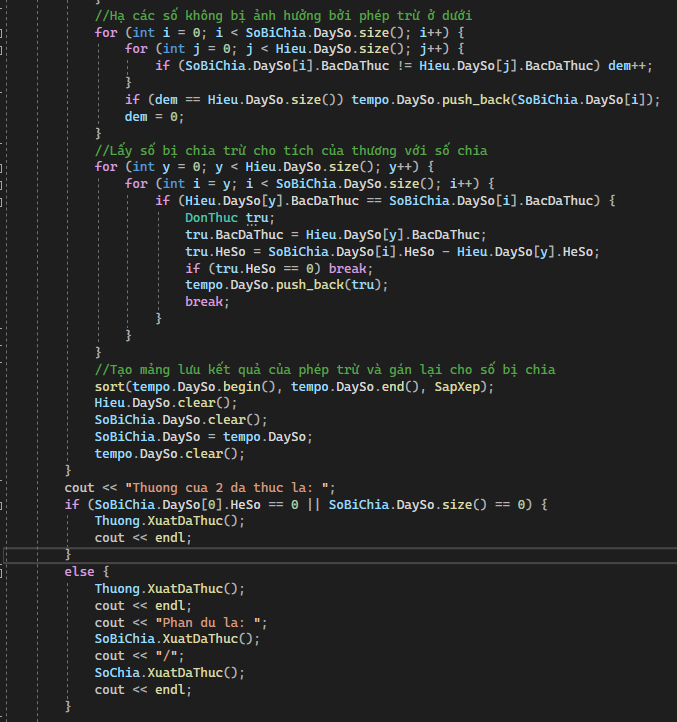


Hàm nhân hai đa thức cần truyền vào thêm một biến có kiểu dữ liệu là lớp DaThuc. Ta thực hiện phép nhân hai đa thức bằng cách lần lượt lấy phần tử của đa thứ nhất nhân với toàn bộ phần tử của đa thức thứ hai, ta nhân hệ số với nhau và cộng số bậc lại, làm liên tục cho đến khi cuối mảng thứ nhất thì dừng và đồng thời truyền kết quả của phép nhân sang một mảng mới. Sau đó ta dùng đến hàm sort trong thư viện algorithm kèm với hàm SapXep đã được định nghĩa ở trên để sắp xếp lại mảng kết quả theo thứ tự giảm dần về số bậc. Sau đó ta tiến hành rút gọn lại đa thức bằng cách cộng (hoặc trừ) tất cả hệ số có cùng số bậc với nhau lại. Cuối cùng truyền kết quả việc rút gọn đa thức sang mảng mới và gọi đến hàm xuất để xuất ra kết quả ngoài màn hình.

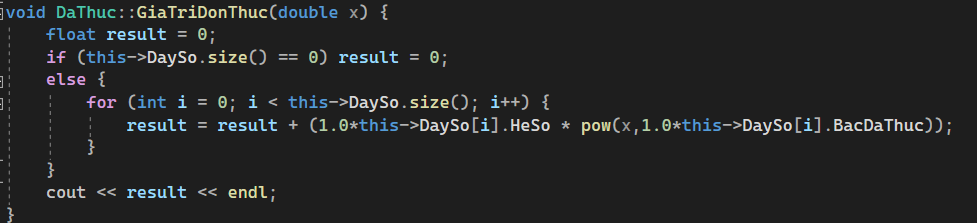


Hàm chia 2 đa thức sẽ truyền vào thêm một biến có kiểu dữ liệu là lớp DaThuc. Đầu tiên ta kiểm tra nên số bậc cao nhất của đa thức 1 nhỏ hơn số bậc cao nhất của đa thức 2 hoặc hệ số của số bậc cao nhất của đa thức 1 nhỏ hơn đa thức 2 thì ta tiến hành xuất ra đa thức 1 chia đa thức 2 (vì không thực hiện được phép chia đa thức). Còn nếu không thì ta tiến hành lấy hệ số của đa thức có số bậc cao nhất của đa thức 1 chia cho hệ số đó của đa thức 2, còn số bậc của 2 đa thức thì trừ. Ra được kết quả ta truyền vào mảng chứa thương của phép chia. Rồi ta nhân ngược kết quả và lấy đa thức 1 trừ đi thì ta sẽ cập nhật lại đa thức 1. Cứ tiếp tục làm như trên cho đến khi hai đa thức không thể chia được thì dừng và xuất ra thương cũng như dư của phép chia. Sau đó gọi đến hàm xuất để xuất ra thương và phép dư của phép chia 2 đa thức

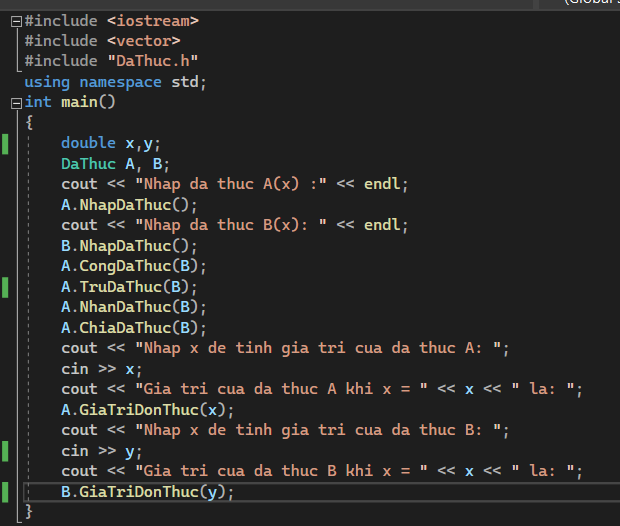




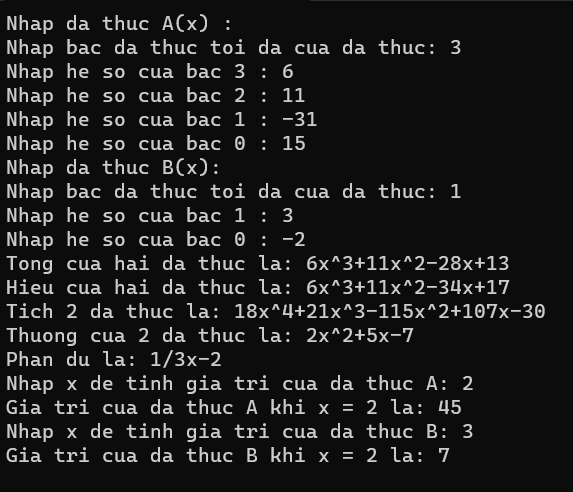
Đến hàm thế giá trị bất kỳ vào đa thức thì ta tiến hành yêu cầu người dùng nhập vào giá trị của biến x trong đa thức để tính. Trong hàm ta sẽ lần lượt cho giá trị mũ giảm dần nhân với hệ số theo mũ giảm dần. Cuối cùng ta xuất ra kết quả của phép thế x vào trong đa thức.



* Bước 3: Tạo thêm 1 file C++ và kèm theo các thư viện cần thiết như iostream, vector, và file header vừa tạo ở trên. Trong hàm main ta lần lượt khai báo hai biến A, B có kiểu dữ liệu trả về là lớp DaThuc, sau đó lần lượt gọi đến hàm nhập đa thức, hàm cộng, trừ, nhân, chia đa thức và hàm truyền giá trị của x vào đa thức để tính giá trị đa thức.



Ví dụ demo:



1. Ghi nội dung công việc, các kịch bản trong bài Thực hành [↑](#footnote-ref-1)