Câu 1: a) Nếu hàm phanso đúng, thì:

Sau khi khai báo, a = 1/3, b = 8

Dòng 27 c = a + (phanso)b; nghĩa là chuyển b từ số nguyên (int) sang dạng phân số để có thể + 2 phân số

c = 25/3

dòng 28 in ra “c = 25/3 + 8 = 25/3”

dòng 29 và 30 in ra trên 1 line: “++c:c = 28/3”. c được tăng lên 1 đơn vị (++)

dòng 31 in ra “a = 1/3 + 28/3”

dòng 32 tăng a thêm c, tức là a = a + c = 29/3

dòng 33 in ra “a = 29/3”

dòng 34 gán giá trị b = a = 29/3, nhưng vì b thuộc kiểu int, a là kiểu PhanSo, nó sẽ ép kiểu PhanSo thành int, trở thành 29/3=9,66666(7), nhưng chỉ lấy phần nguyên nên b = 9

dòng 35 in ra “b = 9”

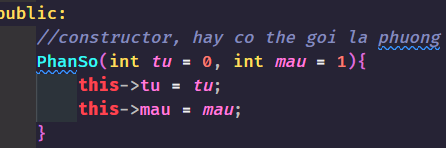
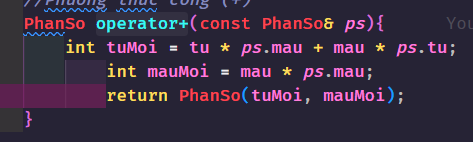
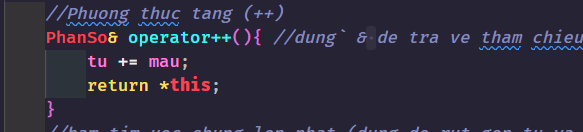
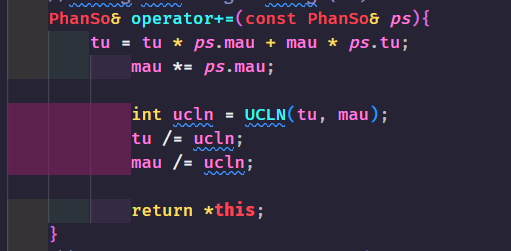
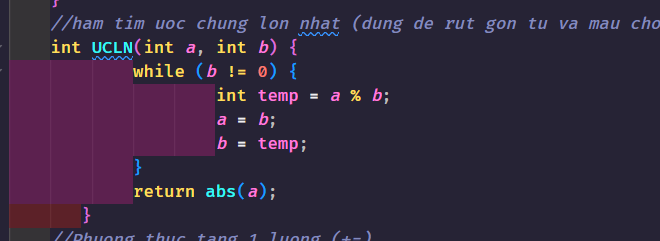
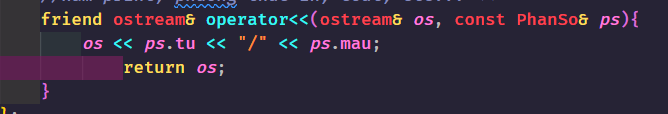
*Minh họa output:*

A black background with white text

Description automatically generated

b) file 2012C1.cpp và head là PhanSo.h  
  
hàm ép kiểu dùng operator+ a + (Phanso)b  
 operator tăng +cc  
 operator tăng một lượng +=   
 operator bang b = a là ép kiểu của thăng a về int để bằng b

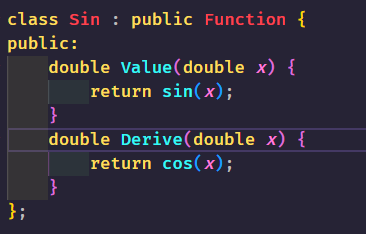
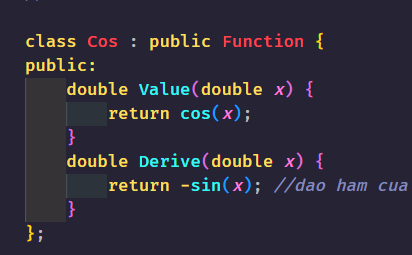
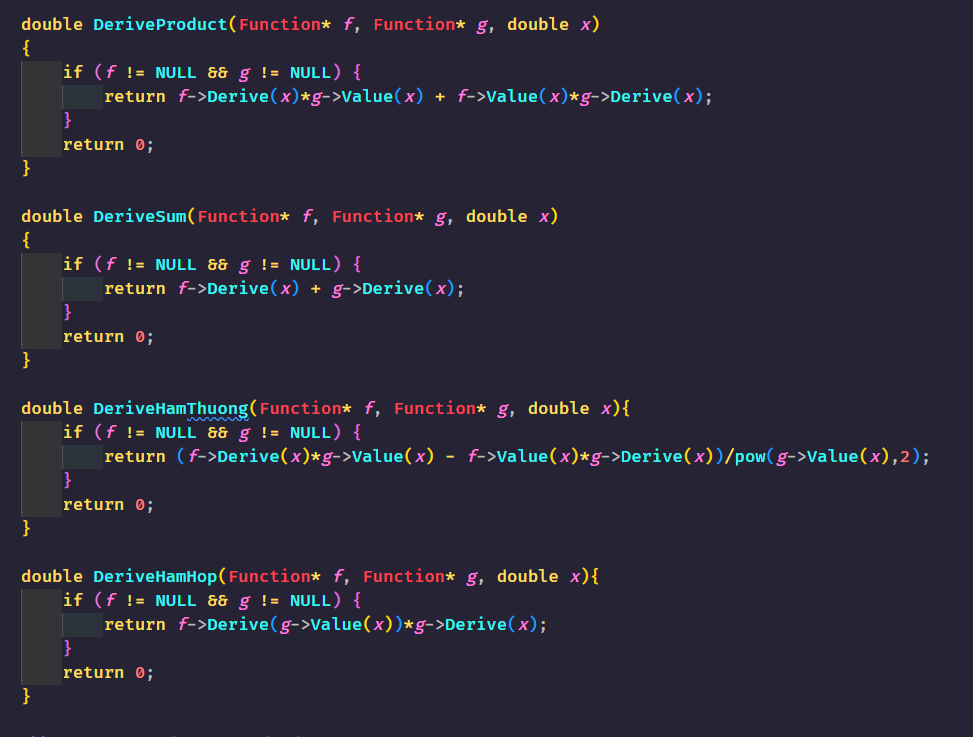
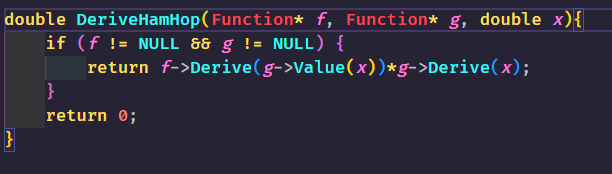
classPhanSo khai báo 2 biến tu,mau  
thứ nhất con constructor có thể dùng để khởi tạo hoặc dùng ép kiểu cho

ép kiểu từ số nguyên thành phân số   
  
khi khởi tạo: “ tu ” trong private gán vào “int tu = 0 “ và mau gán vô “int mau = 1”  
luôn luôn khác không thì mẫu để 1   
- nếu muốn kĩ nữa thì ta xét if(mau!=0) | this ->mau = mau;  
  
khi mình dùng const khi con số được gọi vào trong operator+ thì biến const biến không thể thay đổi nếu trong đoạn dưới chạy có ảnh hưởng tới biến const Phanso thì nó vẫn giữ nguyên giá trị không đổi  
- Trong biến operator+ mình gọi 2 biến mới là “tuMoi” ps.mau là thằng phân số truy cập tới mẫu, tương tự ps.tu cũng vậy truy cập tới tử, còn giữ các dấu nhân là quy đồng mẫu số và “mauMoi”  
Sau đó nó trả về một phân số mới  
- **Operator++** **dùng** dấu & ( và) để trả về tham chiếu thay vì bản sao để khỏi tạo ra nhiều biến mới  
vd như 1/3 + 1 =   
1/3 + 3/3  
  
**Operator+=** là   
vd 1/3 += 2 tăng 1 lượng bằng 2 thì 2 ép thành 1 dàng phân số => 1/3 += 2/1 cái tử bằng là 1/3 + 2/1 ( quy đồng mẫu số ) = 1+6/3  
Khi mình gọi operater tăng nhiều khi tăng một lương lớn quá vd 27/3 chưa rút gọn   
thi phải viết tham một cái hàm   
  
ước chung lớn nhất đã học cấp 2  
thì nó ước chung lớn nhất 27/3 là 3 , 27/3 là 9, 3/3 là 1 => 9/1 là 9   
  
operator này dùng để / cho mấy hàm cout <<   
c) đủ 5 phương thức trong PhanSo.h đã code

Câu 2: Câu này lên mạng chép rõ ràng đầy đủ hơn, ở đây mình nói đơn giản + cho ví dụ thôi

1. Đa hình cho phép một đối tượng hoạt động như một đối tượng của lớp khác, kiểu như có 1 lớp Động Vật, có hàm tạo ra tiếng kêu của nó; một lớp Mèo kế thừa lại lớp Động Vật, có một tiếng kêu riêng “meo meo”, ta có thể dùng đa hình để ghi đè (override) tiếng kêu của Mèo lên động vật. chủ yếu để tránh nhầm lẫn thì tùy theo con trỏ nào được gọi, nó sẽ lấy phương thức của lớp đó. Đây chính là ví dụ cơ bản của đa hình. Ví dụ code trong file 2012C2.cpp
2. Lớp trừu tượng là lớp không thể tạo ra đối tượng từ nó nhưng có thể được kế thừa bởi các lóp khác. Trong nó chứa một hoặc nhiều phương thức trừu tượng (phương thức không có body).

Câu 3:

1. Đầu tiên, class Function là một lớp trừu tượng, với 2 phương thức ảo là Value(x) và Derive(x) không có body (được gán = 0). Mang ý nghĩa như câu 2b, và 2 hàm này bắt buộc class kế thứa Function phải cung cấp định nghĩa cho nó. Từ khóa virtual đại diện cho phương thức ảo, và nó có thể được ghi đè trong một lớp dẫn xuất, ở đây là lớp Sin. Thông qua đó ta có thể sử dụng đa hình động, nghĩa là có thể gọi phương thức của lớp dẫn xuất thông qua con trỏ hoặc tham chiếu của lớp cơ sở.
2. File 2012C3.cpp  
   giải thích hàm sin  
      
   Đề cho và kế thừa Function để nó ghi đề 2 cái hàm   
   Value(giá trị): nó sẽ trả về sin(x), thành sin này thuộc về thư viện math và Derive (đạo hàm) trả về cos(x) v  
   
3. File 2012C3.cpp  
     
   f là con trỏ của Function truyền vào, g tương tự   
   khi f trỏ tới Derive(x) sẽ return về giá trị đã được đạo hàm rồi, khi trả về value sẽ trả về giá trị của chính nó   
   F đạo hàm ( f->Derive(x)) G đạo hàm (g-> Derive(x))   
     
     f->Derive(x)\*g->Value(x) + f->Value(x)\*g->Derive(x);  
   Đạm hàm hàm thương   
     
   f->Derive(x)\*g->Value(x) - f->Value(x)\*g->Derive(x))/pow(g->Value(x),2)  
   Đạo hàm hàm hợp:  
     
   f->Derive(g->Value(x))\*g->Derive(x);  
   sau khi in ra thằng biến y và chỉ đạo hàm Product thôi, để mà xem thử và mình làm có đúng ko gọi them hàm y = DeriveSum(s, c, x):  
   có 2 cái delete đê mình giải phòng bộ nhớ.
4. File 2012C3.cpp phần fix dành cho 3d.  
    note hàm sin ban đầu lại và tạo một hàm sin mới   
   class Sin them một con trỏ instance nằm trong private: không thể nào bị ảnh hưởng bới thằng khác ngoài class   
   hàm getInstance(): kiểm tra xem con tro instance được khởi tạo chưa, nếu chưa thì nó sẽ tạo mới một thằng sin, nếu mà khởi tạo rồi thì nó sẽ trả về giá trị duy nhất một giá trị instance thôi  
   sau đó gán instance = 0 , do cái kết thúc hàm nó sẽ tự động xóa con trỏ nên chỗ delete s; bỏ vô note