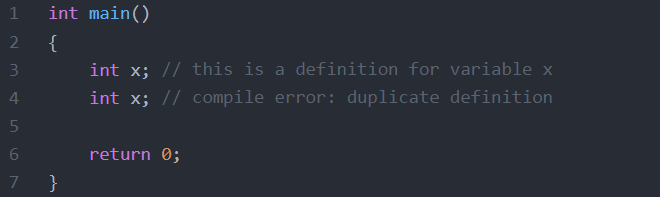
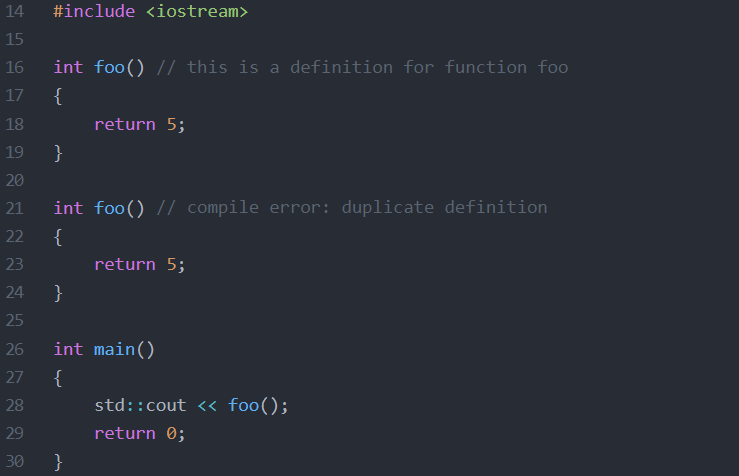
**BẢO VỆ HEADER**

**1. Vấn đề định nghĩa trùng lặp**

Trong bài – Thứ tự khai báo và định nghĩa chuyển tiếp, chúng ta lưu ý rằng một biến định danh hoặc hàm chỉ có thể có một định nghĩa (quy tắc một định nghĩa). Do đó, một chương trình khai báo một ký hiệu nhận dạng biến nhiều làn sẽ gây ra lỗi biên dịch.



Tương tự, các chương trình khai báo một hàm nhiều lần cũng sẽ gây ra lỗi biên dịch.



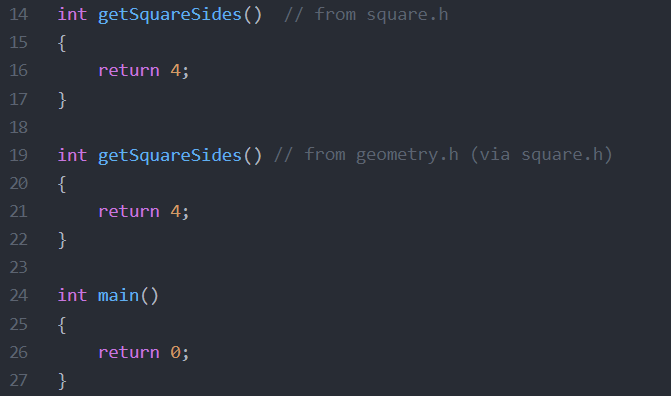
Mặc dù các chương trình này dễ sửa (loại bỏ định nghĩa trùng lặp), với các file heder, khá dễ dẫn đến tình huống một định nghĩa trong file header được đưa vào nhiều lần. Điều này có thể xảy ra khi file header #include file header khác (điều này thường xảy ra).

Hãy xem xét ví dụ hoặc thuật sau:



Chương trình trông có vẻ đơn giản này sẽ không biên dịch! Đây là những gì đang xảy ra. Đầu tiên, main.cpp #include square.h, sao chép định nghĩa cho hàm getSquareSides vào main.cpp. Sau đó, main.cpp #include geometry.h, chính là geometry file #include square.h. Thao tác này sao chép nội dung square.h (bao gồm cả định nghĩa cho hàm getSquareSides) vào geometry.h, sau đó được sao chép vào main.cpp.

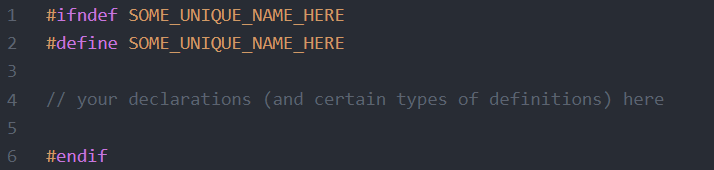
Do đó, sau khi giải quyết tất cả #includes, main.cpp trông như thế này.



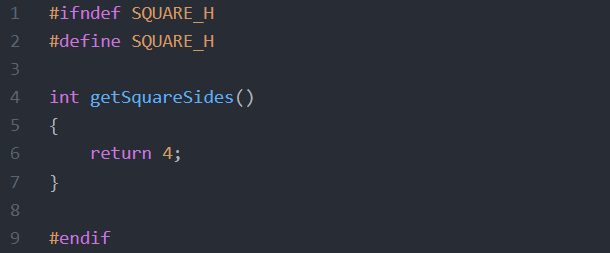
Định nghĩa trùng lặp và lỗi biên dịch. Mỗi file, riêng lẻ, đều tốt. Tuy nhiên, vì main.cpp kết thúc #including nội dung của square.h hai lần, chúng ta đã gặp sự cố. Nếu geometry.h cần getSquareSides() và main.cpp cần cả geometry.h và square.h, bạn sẽ giải quyết vấn đề này như thế nào?

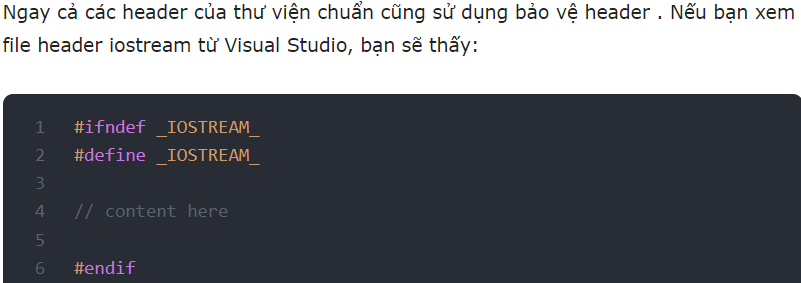
**2. Bảo vệ header**

Tin tốt là chúng ta có thể tránh được vấn đề trên thông qua một cơ chế được gọi là bộ bảo vệ header (còn gọi là bộ bảo vệ include). Bảo vệ Header là các lệnh biên dịch có điều kện có dạng sau:



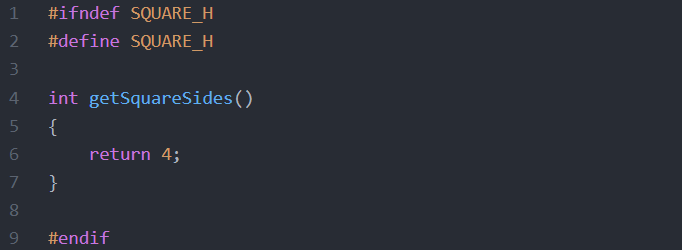
Tất cả các file header của bạn phải có bảo vệ header trên chúng. SOME\_UNIQUE\_NAME\_HERE có thể là bất kỳ tên nào bạn muốn, nhưng theo quy ước đặt thành tên file đầy đủ của file header, được nhập toàn bộ bằng chữ hoa, sử dụng dấu gạch dưới cho khaongr trắng hoặc dấu câu. Ví dụ: square.h sẽ có phần bảo vệ header:

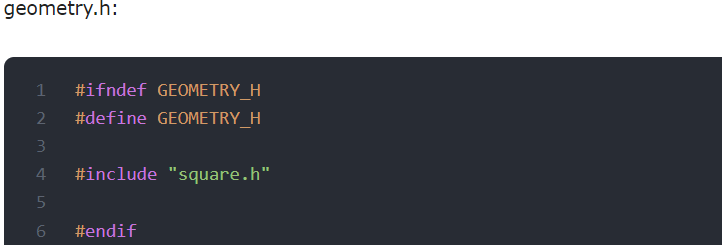




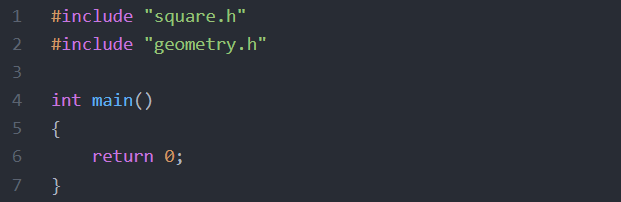
**Cập nhật ví dụ trước của chúng ta với bộ bảo vệ header:**

Hãy quay lại ví dụ square.h, sử dụng square.h với các bộ bảo vệ header. Để có hình thức tốt, chúng ta cũng sẽ thêm các phần bảo vệ header cho geometry.h





main.cpp



Sau khi bộ xử lý tiền xử lý giải quyết tất cả các include, chương trình này trông giống như sau:

main.cpp

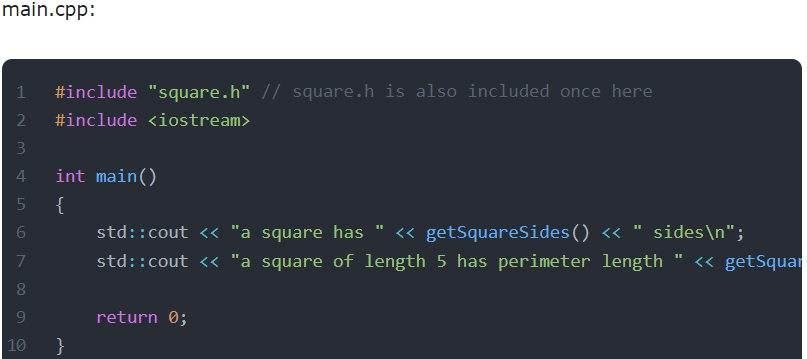


Như bạn có thể thấy từ ví dụ, phần bao gồm nội dung thứ hai của square.h (từ geometry.h) bị bỏ qua vì SQUARE\_H đã được đưa khai báo từ phần include đầu tiên. Do đó, hàm getSquareSides chỉ được đưa vào một lần.

Trình bảo vệ header không ngăn header được include một lần vào các file code khác nhau

Lưu ý rằng mục tiêu của trình bảo vệ header là ngăn một file code nhận nhiều hơn một bản sao của header file được bảo vệ. Theo thiết kế, bộ bảo vệ header không ngăn một file header nhất định được đưa vào (một lần) vào các file code riêng biệt. Điều này cũng có thể gây ra sự cố không mong muốn. Xem xét:



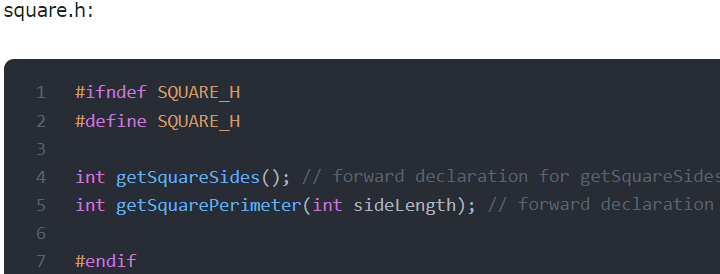


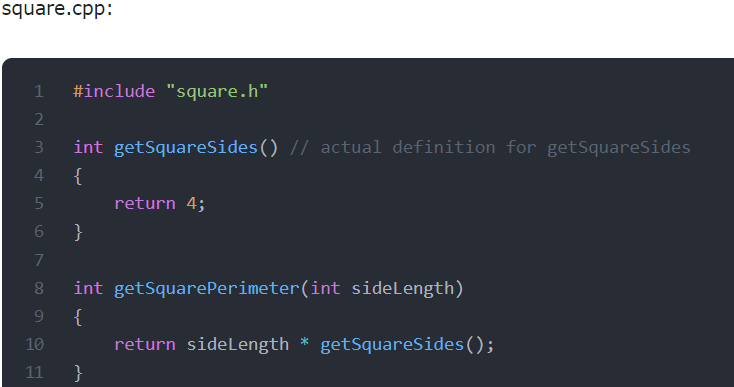
Lưu ý rằng square.h được include từ cả main.cpp và square.cpp. Điều này có nghĩa là nội dung của square.h sẽ được đưa một lần vào square.cpp và một lần vào main.cpp.

Hãy xem xét lý do tại sao điều này xảy ra chi tiết hơn. Khi square.h được include từ square.cpp, SQUARE\_H được khai báo cho đến cuối file square.cpp. Định nghĩa này ngăn không square.h được đưa vào square.cpp lần thứ hai (là điểm của các trình bảo vệ header). Tuy nhiên, khi square.cpp kết thúc, SQUARE\_H không còn được coi là đã định nghĩa. Điều này có nghĩa là khi bộ tiền xử lý chạy trên main.cpp, SQUARE\_H ban đầu không được định nghĩa trong main.cpp.

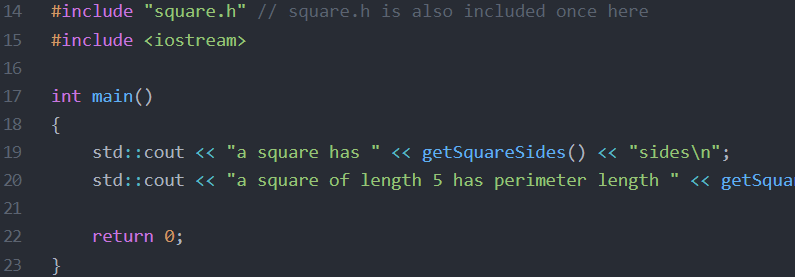
Kết quả cuối cùng là cả square.cpp và main.cpp đều nhận được bản sao định nghĩa của getSquareSides. **Chương trình này sẽ biên dịch, nhưng trình liên kết sẽ phàn nà về việc chương trình của bạn có nhiều định nghĩa cho getSquareSides được nhận dạng!.**

Cách tốt nhất để giải quyết vấn đề này chỉ đơn giản là **đặt định nghĩa hàm vào một trong các file .cpp để header chỉ chứa một khai báo chuyển tiếp**.





main.cpp



Bây giờ khi chương trình được biên dịch, hàm getSquareSides sẽ chỉ có một định nghĩa (thông qua square.cpp), vì vậy trình liên kết rất vui. file main.cpp có thể gọi hàm này (mặc dù nó sống trong square.cpp) vì nó bao gồm square.h, có khai báo chuyển tiếp cho hàm (trình liên kết sẽ kết nối lệnh gọi getSquareSides từ main.cpp với định nghĩa của getSquareSides trong square.cpp).

**3. Chúng ta có thể tránh các định nghĩa trong file header không?**

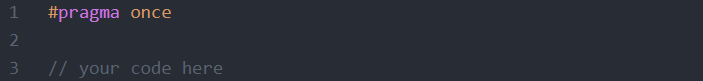
Nói chung, chúng ta đã nói với bạn rằng không bao gồm các định nghĩa hàm trong header file của bạn. Vì vậy, bạn có thể tự hỏi tại sao bạn nên include các bảo vệ header nếu chúng bảo vệ bạn khỏi những điều bạn không nên làm.

Trong tương lai, chúng ta sẽ chỉ cho bạn khá nhiều trường hợp cần phải đưa các định nghĩa hàm vào file header. Ví dụ C++ sẽ cho phép bạn tạo các kiểu riêng mình. Các kiểu do người dùng định nghĩa này thường được khai báo trong các file header, vì vậy định nghĩa có thể được truyền ra các file code cần sử dụng. Nếu không có bộ bảo vệ header, các file code của bạn có thể có nhiều bản sao giống hệt nhau của các định nghĩa này, điều này sẽ gây ra lỗi biên dịch định nghĩa trùng lặp.

Vì vậy, mặc dù không nhất thiết phải có bảo vệ header vào thời điểm này trong loạt bài hướng dẫn, nhưng chúng ta đang thiết lập những thói quen tốt ngay bây giờ, vì vậy bạn không phải bỏ học những thói quen xấu sau này.

**4. #pragma một lần**

Nhiều trình biên dịch hỗ trợ một dạng bảo vệ header thay thế, đơn giản hơn bằng cách sử dụng chỉ thị #pragma:



#pragma từng phục vụ cùng mục đích như trình bảo vệ header và có thêm lợi ích là ngắn hơn và ít mắc lỗi hơn.

Tuy nhiên, #pragma một lần không phải là một phần chính thức của ngôn ngữ C++ và không phải tất cả các trình biên dịch đều hỗ trợ nó (mặc dù hầu hết các trình biên dịch hiện đại đều có).

Vì mục đích tương thích, chúng ta khuyên bạn nên sử dụng các trình bảo header truyền thống. Chúng không còn nhiều công việc nữa và chúng được đảm bảo sẽ được hỗ trợ trên tất cả các trình biên dịch tuân thủ.