**HẰNG SỐ, CONSTEXP VÀ HẰNG SỐ TƯỢNG TRƯNG**

**1. Biến Const**

Để tạo một biến không đổi, chỉ cần đặt từ khóa const trước hoặc sau kiểu dữ liệu của biến, như sau:

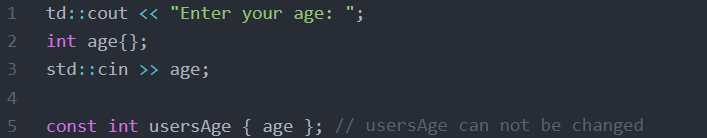


Mặc dù C++ sẽ chấp nhận const trước hoặc sau kiểu dữ liệu, nhưng chúng tôi khuyên bạn nên sử dụng const trước kiểu dữ liệu vì nó tuân theo quy ước ngôn ngữ tiếng Anh tiêu chuẩn tốt hơn trong đó các bổ ngữ đến trước đối tượng được sửa đổi.

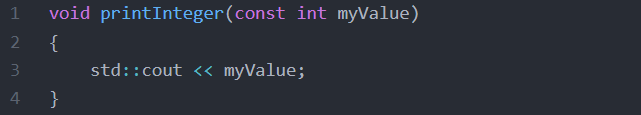
Biến const phải được khởi tạo khi bạn định nghĩa chúng, và sau đó giá trị đó không thể thay đổi thông qua phép gán.

Khai báo một biến const ngăn chúng ta vô tình thay đổi giá trị của nó.

**Lưu ý rằng các biến const có thể được khởi tạo từ các biến khác (bao gồm cả những biến không phải const):**



Const thường được sử dụng với các tham số hàm:



Tạo một tham số hàm thực hiện 1 điều. Nó cho người gọi hàm biết rằng hàm sẽ không thay đổi giá trị của myValue

Thời gian chạy với hằng số thời gian biên dịch C++ thực sự có hai loại hằng số khác nhau.

**Hằng số thời gian chạy** là những hằng số có giá trị khởi tạo chỉ có thể được sử dụng trong thời gian chạy (khi chương trình của bạn đang chạy). Các biến như usersAge và myValue trong các đoạn code ở trên là các hằng số thời gian chạy, vì trình biên dịch không thể xác định giá trị ban đầu của chúng tại thời điểm biên dịch. userAge dựa vào đầu vào của người dùng (chỉ có thể được cung cấp trong thời gian chạy) và myValue phụ thuộc vào giá trị được truyền vào hàm (chỉ được biết trong thời gian chạy). Tuy nhiên, sau khi khởi tạo, giá trị của hằng số này không thể thay đổi.

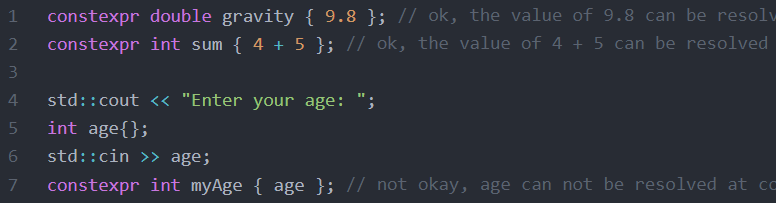
Hằng số thời gian biên dịch là những hằng số mà các giá trị khởi tạo có thể được sử dụng tại thời điểm biên dịch (khi chương trình của bạn đang biên dịch). Trọng lực biến đổi ở trên là một ví dụ về hằng số thời gian biên dịch. Hằng số thời gian biên dịch cho phép trình biên dịch thực hiện các tối ưu hóa mà hằng số thời gian chạy không có sẵn. Ví dụ, bất cứ khi nào cần được sử dụng, trình biên dịch có thể chỉ cần thay thế tên định danh bằng chữ double 9,8.

**Khi bạn khai báo một biến const, trình biên dịch sẽ ngầm theo dõi xem đó là hằng số thời gian chạy hay thời gian biên dịch.**

Trong hầu hết các trường hợp, điều này không thành vấn đề, nhưng có một vài trường hợp kỳ lạ trong đó C++ **yêu cầu hằng số thời gian biên dịch thay vì hằng số thời gian chạy,** ví dụ: trong phần khởi tạo kiểu – điều mà chúng ta sẽ đề cập sau.

**2. constexpr**

Để cung cấp tính cụ thể hơn, C++ 11 đã giới thiệu từ khóa constexpr, đảm ảo rằng hằng số phải là hằng số thời gian biên dịch:



**Biến constexpr là const**. Điều này sẽ trở nên quan trọng khi chúng ta nói về các tác dụng của const trong các bài học sắp tới.

**Bạn nên – Bất kỳ biến nào không thể sửa đổi được sau khi khởi tạo và có trình khởi tạo được biết đến tại thời điểm biên dịch nên được khai báo là constexpr.**

**Bất kỳ biến nào không thể sửa đổi sau khi khởi tạo và có bộ khởi tạo không được xác định tại thười điểm biên dịch nên được khai báo là const.**

**3. Đặt tên cho các biến const của bạn**

Một số lập trình viên thích sử dụng tất cả các tên viết hoa cho các biến const. Những người khác sử dụng tên biến bình thường với tiền tố ‘k’. Tuy nhiên, chúng ta sẽ sử dụng các quy ước đặt tên biến thông thường, phổ biến hơn. Các biến Const hoạt động chính xác như các biến bình thường trong mọi trường hợp ngoại trừ việc chúng không thể được gán cho, vì vậy không có lý do cụ thể nào mà chúng cần được biểu thị là đặc biệt.

Lưu ý – Tránh sử dụng #define để tạo macro hằng số tượng trưng.

Một giải pháp tốt hơn: Sử dụng các biến constexpr

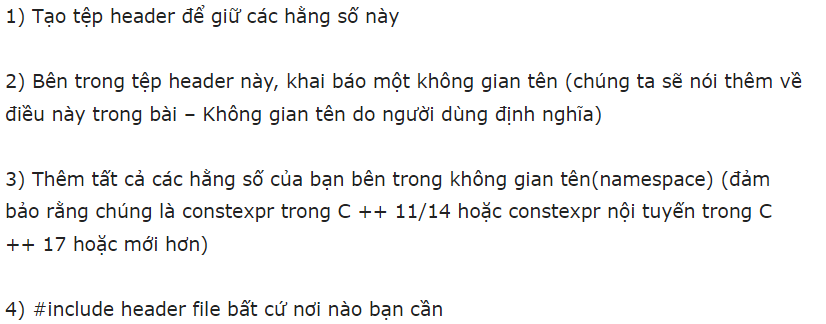


Vì đây chỉ là các biến bình thường nên chúng có thể xem được trong trình gỡ lỗi, có phạm vi bình thường và tránh các hành vi kỳ lạ khác.

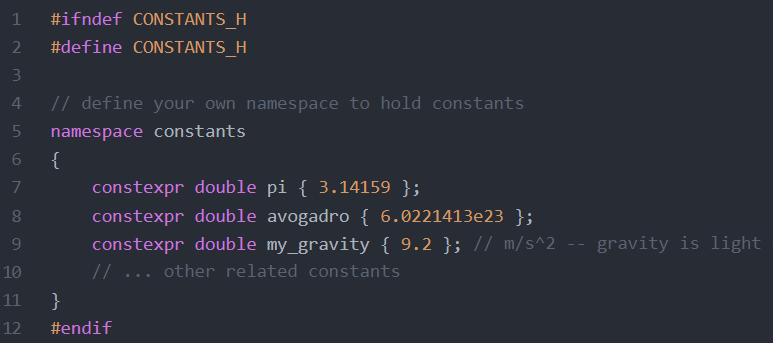
**Sử dụng hằng số tượng trưng trong một chương trình nhiều file.**

Trong nhiều ứng dụng, một hằng số tượng trưng nhất định được sử dụng trong toàn bộ code của bạn (không chỉ ở một vị trí). Chúng có thể bao gồm các hằng số vật lý hoặc toán học không thay đổi (ví dụ: số pi hoặc số của avogadro) hoặc các giá trị “điều chỉnh” dành riêng cho ứng dụng (ví dụ: hệ số ma sát hoặc trọng lực). Thay vì khai báo lại chúng mỗi khi cần, tốt hơn nên khai báo chúng một lần ở vị trí trugn tâm và sử dụng chúng ở bất cứ nơi nào cần thiết. Bằng cách đó, nếu bạn cần thay đổi chúng, bạn chỉ cần thay chúng ở một nơi.

Có nhiều cách để hỗ trợ điều này trong C++, nhưng cách sau có lẽ là dễ nhất:

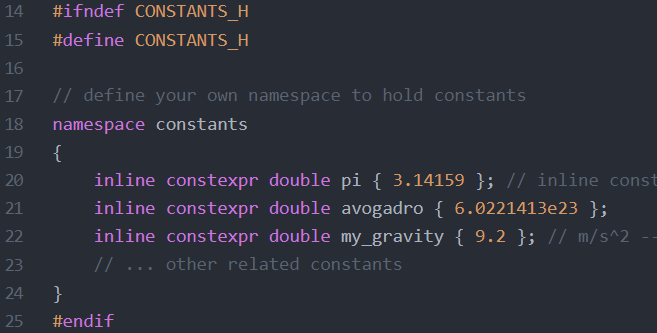


Ví dụ: constants.h (C++ 11/14):

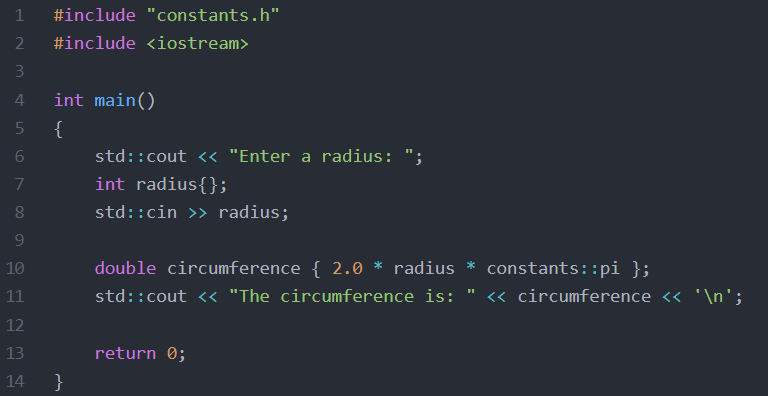


Trong C++ 17, ưu tiên “inline constexpr” thay thế:

Constants.h (C++ 17 hoặc mới hơn):



Sử dụng toán tử phân giải phạm vi để truy cập các hằng số của bạn trong tệp .cpp



Nếu bạn có cả hằng số vật lý và gia trị điều chỉnh cho mỗi ứng dụng, bạn có thể chọn sử dụng hai file – một cho các giá trị vật lý sẽ không bao giờ thay đổi và một cho các giá trị điều chỉnh cho mỗi chương trình cụ thể cho chương trình của bạn. Bằng cách đó, bạn có thể sử dụng lại các giá trị vật lý trong bất kỳ chương trình nào khác.