**GIỚI THIỆU VỀ KIỂU DỮ LIỆU CHAR TRONG C++**

Ký tự char luôn được đặt giữa các dấu nháy đơn.

**1. Khởi tạo ký tự**

Bạn có thể khởi tạo các biến char bằng cách sử dụng các ký tự:



Bạn cũng có thể khởi tạo ký tự bằng số nguyên, nhưng điều này nên tránh nếu có thể:

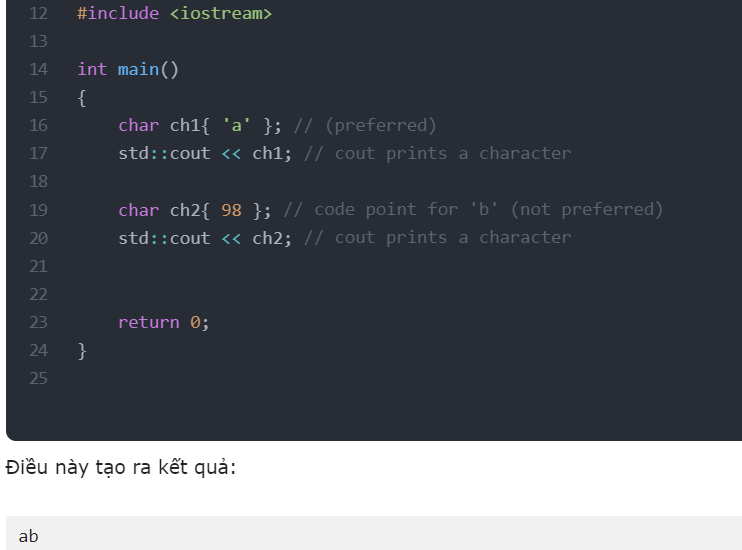


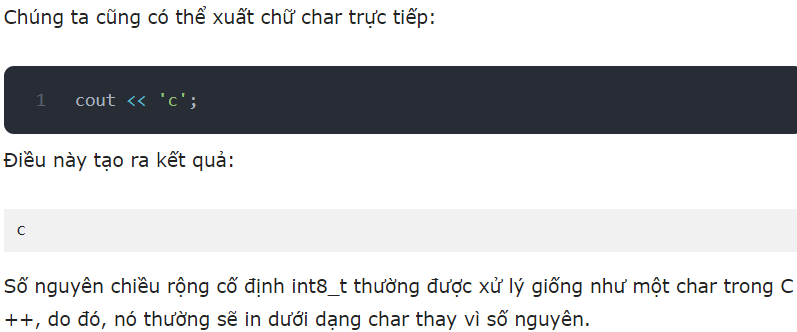
Cẩn thận không trộn lẫn số ký tự với số nguyên. Hai khởi tạo sau không giống nhau:



Số dưới dạng ký tự sẽ được sử dụng khi chúng ta muốn biểu diễn số dưới dạng văn bản, thay vì số để áp dụng các phép toán.

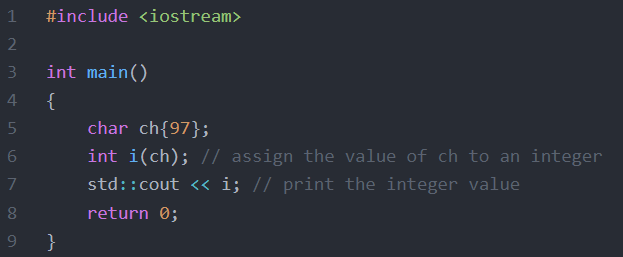
**In ký tự:**



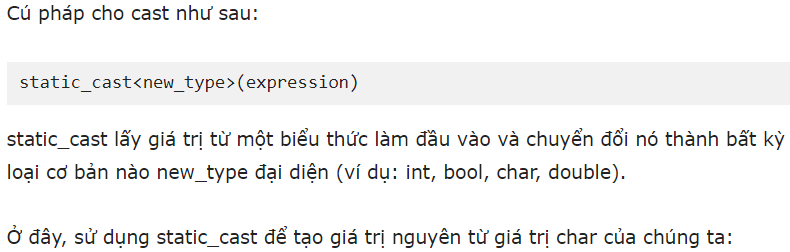


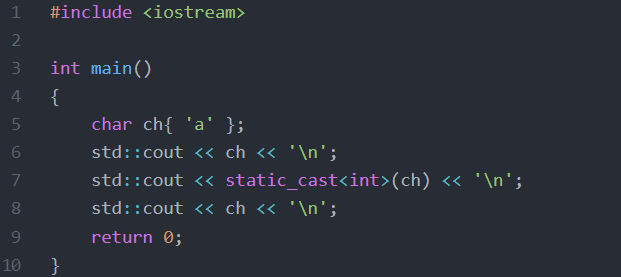
**2. In ký tự dưới dạng số nguyên thông qua static\_cast**

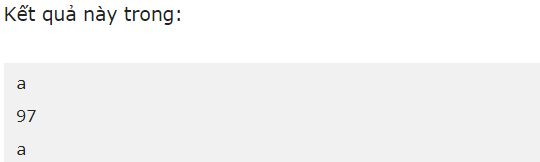
Nếu chúng ta muốn xuất một char dưới dạng một số thay vì một ký tự, chúng ta phải nói với std::cout để in char như thể nó là một số nguyên. Một cách để làm điều này là gán char cho một số nguyên và in số nguyên:



Tuy nhiên, điều này là khó khăn. Một cách tốt hơn là sử dụng một kiểu cast. Một cast sẽ tạo ra một giá trị cho một kiểu nào đó từ một giá trị của kiểu khác. Để chuyển đổi giữa các kiểu dữ liệu cơ bản (ví dụ: từ char thành int hoặc ngược lại), chúng ta sử dụng static\_cast.

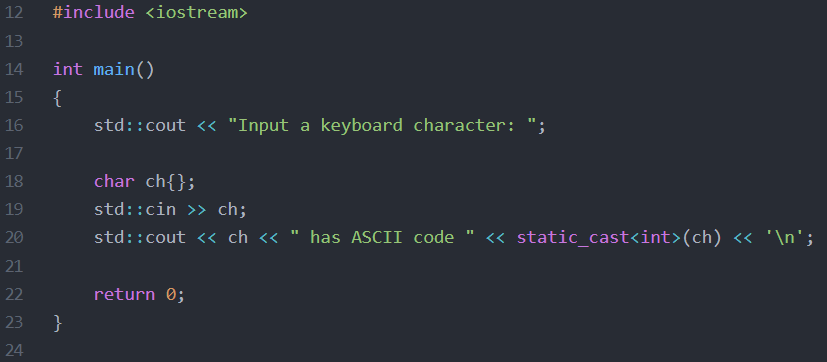


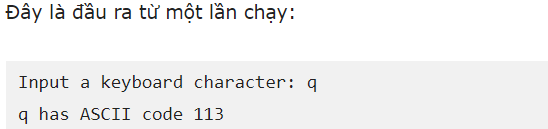




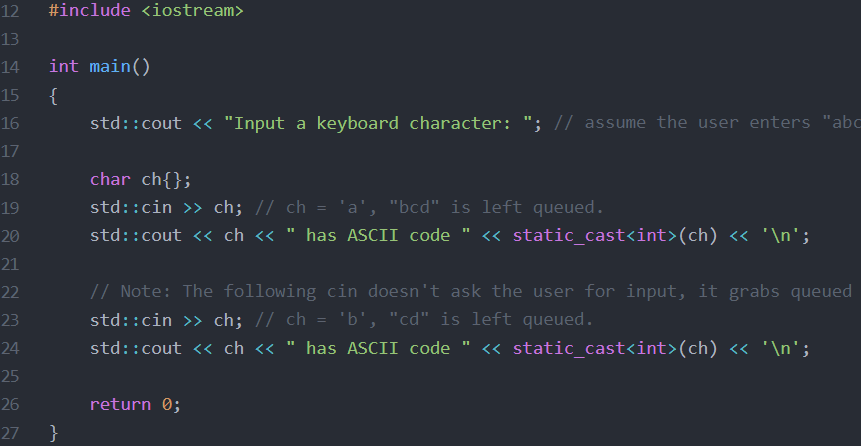
**3. Nhập ký tự**

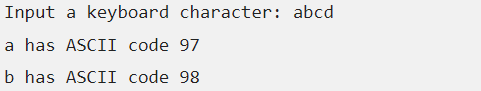
Chương trình sau đây yêu cầu người dùng nhập một ký tự, sau đó in ra cả ký tự và code ASCII của nó:





Lưu ý rằng std::cin sẽ cho phép bạn nhập nhiều ký tự. Tuy nhiên, biến ch chỉ có thể chứa một 1 kỳ. Do đó, chỉ có ký tự đầu vào đầu tiên được trích xuất thành biến ch. Phần còn lại của đầu vào người dùng để lại trong bộ đệm đầu vào mà std::cin sử dụng và có thể được trích xuất bằng các lệnh gọi tiếp theo đến std::cin.





**4. Kích thước char, phạm vi và dấu hiệu mặc định**

Char được định nghĩa bởi C ++ để luôn có kích thước 1 byte. Theo mặc định, một char có thể được có dấu hoặc không dấu (mặc dù nó thường được có dấu). Nếu bạn sử dụng ký tự để giữ các ký tự ASCII, bạn không cần chỉ định một ký hiệu (vì cả hai ký tự có ký hiệu và không dấu có thể giữ các giá trị trong khoảng từ 0 đến 127).

Nếu bạn sử dụng char để giữ các số nguyên nhỏ (điều bạn không nên làm trừ khi bạn tối ưu hóa rõ ràng cho không gian), bạn phải luôn chỉ định xem nó có dấu hay không dấu. Một char có dấunhân vật có thể giữ một số trong khoảng từ -128 đến 127. Một char không dấu có thể giữ một số trong khoảng từ 0 đến 255.

**5. Trình tự thoát khỏi một dòng nào đó**



**6. Điều khác biệt giữa việc đặt ký hiệu trong dấu ngoặc đơn hay dấu ngoặc kép**

Các ký tự độc lập luôn được đặt trong các trích dẫn đơn (Ví dụ: ‘a’ + ‘5’). Một char chỉ có thể đại diện cho một ký hiệu (ví dụ: chữ a, ký hiệu cộng, số 5). Ví dụ sau đây là sai:



Văn bản đặt giữa hai dấu ngoặc kép (ví dụ: “hello, world!”) được gọi là một chuỗi. Chuỗi là một tập hợp các ký tự liên tiếp (Và do đó, một chuỗi có thể chứa nhiều ký hiệu).

Hiện tại, bạn có thể sử dụng chuỗi ký tự trong code của mình.



**Tuy nhiên, các chuỗi không phải là kiểu dữ liệu cơ bản trong C++ và nó phức tạp hơn một chút, vì vậy chúng ta sẽ bảo lưu thảo luận về các kiểu dữ liệu đó cho đến khi chúng tôi đề cập đến các kiểu dữ liệu ghép.**

Luôn đặt các ký tự độc lập trogn các dấu ngoặc đơn. Điều này giúp trình biên dịch tối ưu hóa hiệu quả hơn.

**8. Còn các loại char kahcs, wchar\_t, char16\_t và char32\_t thì sao?**

Nên tránh sử dụng wchar\_t trong hầu hết các trường hợp (trừ khi giao tiếp với API Windows). Kích thước của nó được xác định, và không đáng tin cậy. Nó phần lớn đã bị phản đối khi dùng.

Giống như ASCII ánh xạ các số nguyên 0-127 sang các ký tự tiếng US, các tiêu chuẩn code hóa ký tự khác tồn tại để ánh xạ các số nguyên (có kích thước khác nhau) sang các ký tự trong các ngôn ngữ khác. Ánh xạ nổi tiếng nhất bên ngoài ASCII là tiêu chuẩn Unicode, ánh xạ hơn 110.000 số nguyên sang các ký tự trong nhiều ngôn ngữ khác nhau. Vì Unicode chứa rất nhiều điểm code, một điểm code Unicode duy nhất cần 32 bit để thể hiện một ký tự (được gọi là UTF-32). Tuy nhiên, các ký tự Unicode cũng có thể được mã hóa bằng nhiều ký tự 16 bit hoặc 8 bit (được gọi là UTF-16 và UTF-8 tương ứng).

char16\_t và char32\_t đã được thêm vào C ++ 11 để cung cấp hỗ trợ rõ ràng cho các ký tự Unicode 16 bit và 32 bit. char8\_t đã được thêm vào trong C ++ 20.

Bạn có nhu cầu sử dụng char8\_t, char16\_t hoặc char32\_t trừ khi bạn có kế hoạch làm một chương trình có liên quan tới Unicode. Unicode và ngôn ngữ bản địa nói chung nằm ngoài phạm vi của các hướng dẫn này, vì vậy chúng ta sẽ tìm hiểu trong phần tiếp theo.

Trong khi đó, bạn chỉ nên sử dụng các ký tự ASCII khi làm việc với các ký tự (và chuỗi). Sử dụng các ký tự từ các bộ ký tự khác có thể khiến các ký tự của bạn hiển thị không chính xác.