1. **Tìm hiểu cơ chế lưu trữ biến trong lập trình C:**

**Lớp lưu trữ trong C**được dùng để mô tả các đặc điểm của một biến/hàm. Những tính năng này về cơ bản bao gồm phạm vi, hiển thị và thời gian tồn tại, giúp người dùng theo dõi sự hiện diện của một biến cụ thể trong thời gian chạy một chương trình.

**Ngôn ngữ C** sử dụng 4 class lưu trữ:

* Auto
* Extern
* Static
* Register

Tuy nhiên, chúng có thể được truy cập bên ngoài phạm vi cũng như dùng khái niệm pointer được cung cấp bằng cách trỏ tới vị trí bộ nhớ chính xác tại nơi cư trú các biến. Mặc định, chúng được gắn giá trị rác ở bất kỳ nơi được khai báo.

Ví dụ trên để định nghĩa hai biến trong cùng một lớp lưu trữ, auto có thể được sử dụng bên trong hàm, ví dụ biến cục bộ.

**Lớp lưu trữ auto trong C**

Đây là lớp lưu trữ mặc định cho tất cả các biến được khai báo bên trong một hàm hay một khối. Vì thế, từ khóa auto hiếm khi được dùng lúc viết chương trình. Bạn chỉ có thể truy cập biến auto trong khối/hàm đã được khai báo và không nằm bên ngoài chúng (xác định phạm vi). Tất nhiên, bạn có thể truy cập chúng trong các khối lồng nhau ở khối/hàm cha nơi biến auto được khai báo.

Tuy nhiên, chúng có thể được truy cập bên ngoài phạm vi cũng như dùng khái niệm pointer được cung cấp bằng cách trỏ tới vị trí bộ nhớ chính xác tại nơi cư trú các biến. Mặc định, chúng được gắn giá trị rác ở bất kỳ nơi được khai báo.

## Lớp lưu trữ register trong C

Lớp lưu trữ **register** có thể được sử dụng để định nghĩa biến cục bộ và có thể được lưu trữ trong một vùng đăng ký thay vì RAM. Điều này nghĩa là biến này có cỡ tối đa tương đương với cỡ đăng ký.

Lớp lưu trữ register chỉ được sử dụng cho biến mà yêu cầu truy cập nhanh như bộ đếm. Chú ý rằng định nghĩa 'register' không đồng nghĩa với biến đó có thể lưu trữ trong một thanh ghi. Nó nghĩa là có thể lưu trữ trong thanh ghi phụ thuộc vào phần cứng và với các hạn chế nhất định.

## Lớp lưu trữ static trong C

Lớp lưu trữ**static** hướng dẫn trình biên dịch giữ các giá trị biến**cục bộ** tồn tại trong thời gian sống của chương trình thay vì việc tạo ra và hủy nó mỗi lần chạy qua phạm vi đó. Do đó, tạo một biến **cục bộ** **static** cho phép chúng lưu trữ các giá trị với các hàm gọi.

Lớp static này có thể được áp dụng cho biến toàn cục. Khi việc này diễn ra, nó gây ra phạm vi của biến được giới hạn trong file mà nó khai báo.

Trong lập trình C, khi static được sử dụng ứng với lớp, nó dẫn đến chỉ có một bản copy của lớp khai báo được chia sẻ bởi tất cả các đối tượng sử dụng lớp này.

## Lớp lưu trữ extern trong C

Lóp lưu trữ**extern** được sử dụng để đưa tham chiếu đến các biến toàn cục mà được nhìn thấy bởi tất cả các file chương trình.

Khi bạn có nhiều file và bạn định nghĩa các biến cục bộ hoặc hàm, nó sẽ được sử dụng ở các file khác. Để hiểu vấn đề này, extern được sử dụng để khai báo biến toàn cục hoặc hàm ở file khác.

1. **Tìm hiểu dữ liệu bool và cách sử dụng trong lập trình C:**

* Kiểu dữ liệu Boolean là một kiểu dữ liệu có chỉ có thể nhận một trong hai giá trị như đúng/sai (true/false, yes/no, 1/0) nhằm đại diện cho hai giá trị thật (truth value).
* Trong lập trình C kiểu boolean sẽ được gọi là bool (trong Java thì gọi là boolean, trong Python thì gọi là bool… tùy theo ngôn ngữ).
* Ban đầu, ngôn ngữ C không hỗ trợ kiểu bool, mà nó dùng số integer để biểu thị true/false (0 tức là false, khác 0 tức là true). Bắt đầu từ phiên bản C99 standard for C language thì mới bắt đầu hỗ trợ kiểu bool.
* Để sử dụng kiểu bool ta có các cách sau:

**Cách 1: Sử dụng thư viện <stdbool.h>**

Với cách này ta sử dụng kiểu bool của C

**Cách 2: Tự định nghĩa kiểu bool với enum**

**Cách 3: Định nghĩa kiểu bool với integer**

**Cách 4: Khai báo hằng số true/false**

* Trong các cách trên:
* **Cách 1** chỉ chạy được từ phiên bản C99 standard for C language. Tuy nhiên hầu hết các bản compiler C đều đã update cho nên các bạn nên dùng cách này.
* **Cách 2,3,4**: thì hoạt động giống nhau

Nhìn chung thì kiểu bool vẫn là kiểu integer, nên cho dù dùng cách nào thì bạn vẫn có thể gán bool bằng một giá trị integer bất kì (0 tức là false, khác 0 tức là true).

1. **Tìm hiểu về toán tử 3 ngôi trong lập trình C**

* **Toán tử ba ngôi** (ternary operator) là một trong những khái niệm quan trọng trong lập trình C. Toán tử ba ngôi cho phép thực hiện đồng thời dựa trên một điều kiện và cung cấp một giá trị trả về tùy thuộc vào kết quả **đúng/sai**của điều kiện đó. Toán tử ba ngôi giúp tạo mã ngắn gọn hơn và dễ đọc hơn trong một số trường hợp, ngoài ra nó cũng có thể sử dụng thay thế cho câu lệnh**if else** trong một số trường hợp đơn giản. Trong bài viết này, chúng ta sẽ tìm hiểu cách sử dụng toán tử ba ngôi trong C/C++.
* **Trong đó:**
* **condition**: Là biểu thức điều kiện(Ví dụ: a > 5, b > 5, b<5, a >b….).
* **value\_if\_true**: Gía trị trả về nếu biểu thức điều kiện là **đúng**
* **value\_if\_false**: Giá trị trả về nếu biểu thức điều kiện là **sai**.

=> ***value\_if\_true****,****value\_if\_false***cũng có thể là 1 toán tử ba ngôi cấp 2.

Nói tóm lại biểu thức ba ngôi sẽ trả về hay thực hiện tham số thứ nhất(**sau dấu ?**) nếu đk là đúng, tham số thứ 2(**sau dấu :**) nếu đk là sai.

1. **Tại sao khi đang nhập số mà chuyển sang nhập chuỗi lại bị bỏ qua và lệnh fflush**

* Trong quá trình chạy chương trình ta sẽ phải nhập bằng bàn phím, mọi ký tự bạn gõ vào bàn phím (kể cả ký tự Enter \n) đều được đẩy vào bộ nhớ đệm trước khi được gán vào biến. Nếu trước đó bạn có nhập số bằng scanf hoặc cin, chúng chỉ nhận số chứ không nhận được ký tự Enter, và ký tự Enter vẫn còn trong bộ nhớ đệm. Đến khi nhập chuỗi, các hàm nhập chuỗi nhận được ký tự Enter thì dừng nhập luôn và chương trình vẫn chạy tiếp. Điều này khiến kết quả bị sai.
* Bạn có thể sử dụng các hàm sau để thực hiện xóa bộ nhớ đệm”fflush(stdin)”

+ Hàm fflush() trong thư viện stdio.h cũng có tác dụng tương tự flushall(). Tuy nhiên nó cho phép lựa chọn xóa bộ nhớ đệm cho stream nào. Ở đây ta truyền vào stdin để xóa bộ đệm cho dòng nhập chuẩn, tức là bàn phím.