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

1. Lớp lưu trữ Auto

Lớp lưu trữ auto là lớp lưu trữ mặc định cho tất cả các biến cục bộ.

{

  int mount;

  auto int month;

}

Ví dụ trên định nghĩa hai biến trong cùng một lớp lưu trữ. 'auto' chỉ có thể được sử dụng trong các hàm, tức là các biến cục bộ.

2. Lớp lưu trữ Register

Lớp lưu trữ register được sử dụng để xác định các biến cục bộ cần được lưu trữ trong thanh ghi thay vì RAM. Điều này có nghĩa là biến có kích thước tối đa bằng với kích thước thanh ghi (thường là một từ) và không thể có toán tử đơn nhất áp dụng cho nó (vì nó không có vị trí bộ nhớ).

{

  register int  miles;

}

Thanh ghi chỉ được sử dụng cho các biến yêu cầu truy cập nhanh như bộ đếm. Cũng cần lưu ý rằng việc xác định 'register' không có nghĩa là biến sẽ được lưu trong sổ register mà là nó có thể được lưu trữ trong một thanh ghi nào đó tùy thuộc vào phần cứng và các hạn chế thực hiện.

3. Lớp lưu trữ Static

Lớp lưu trữ static chỉ thị trình biên dịch để giữ một biến cục bộ tồn tại trong suốt thời gian chạy của chương trình thay vì tạo và hủy nó mỗi lần nó đi vào và đi ra khỏi phạm vi.

Lớp lưu trữ Static cũng có thể được áp dụng cho các biến toàn cục. Khi điều này được thực hiện, nó làm cho phạm vi của biến đó bị hạn chế đối với tệp mà nó được khai báo.

Trong lập trình C, khi static được sử dụng trên các lớp, nó chỉ gây ra một bản sao của thành viên đó được chia sẻ bởi tất cả các đối tượng của lớp của nó.

#include

/\* function declaration \*/

void func(void);

static int count = 5; /\* global variable \*/

main() {

  while(count--) {

     func();

  }

  return 0;

}

/\* function definition \*/

void func( void ) {

  static int i = 5; /\* local static variable \*/

  i++;

  printf("i is %d and count is %d\n", i, count);

}

Khi mã trên được biên dịch và thực hiện, nó tạo ra kết quả sau:

i is 6 and count is 4

i is 7 and count is 3

i is 8 and count is 2

i is 9 and count is 1

i is 10 and count is 0

4. Lớp lưu trữ Extern

Lớp lưu trữ extern được sử dụng để cung cấp cho một tham chiếu của một biến toàn cục có thể nhìn thấy tất cả các tệp chương trình. Khi bạn sử dụng 'extern', biến không thể được khởi tạo nó chỉ tên biến tại vị trí lưu trữ đã được xác định trước đó.

Khi bạn có nhiều tệp và bạn xác định một biến toàn cục hoặc hàm thì extern sẽ được sử dụng trong tệp khác để cung cấp tham chiếu về biến hoặc hàm được xác định. Nói cách khác extern được sử dụng để khai báo một biến toàn cục hoặc hàm trong một tệp khác.

Trình sửa đổi bên ngoài thường được sử dụng nhất khi có hai hoặc nhiều tệp chia sẻ cùng các biến hoặc hàm toàn cục như được giải thích bên dưới.

* Tệp đầu tiên: main.c:

#include

int count ;

extern void write\_extern();

main() {

  count = 5;

  write\_extern();

}

* Tệp thứ hai: support.c:

#include

extern int count;

void write\_extern(void) {

  printf("count is %d\n", count);

}

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](https://tuicocach.com/tag/hoc-lap-trinh/) C/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**

* Thông thường, khi nhập một chuỗi trong màn hình console, ta phải có thao tác xóa bộ nhớ đệm bàn phím. Nếu không có thể thấy rằng kết quả nhập chuỗi bị sai hoặc trôi đi mất.
* 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.