# Struct và Union

****1. Struct****

Struct là giải pháp khi bạn cần giải quyết các bài toán thực tế khi mà đối tượng bạn cần lưu lại trong chương trình cần rất nhiều thông tin.

Ví dụ khi bạn muốn lưu thông tin của một sinh viên thì không có kiểu dữ liệu nào trong C phù hợp cả, mà bạn cần tập hợp nhiều kiểu dữ liệu vào làm một.

Struct hay cấu trúc là một kiểu dữ liệu mà tự bạn định nghĩa ra bằng cách gộp nhiều kiểu dữ liệu có sẵn lại nhằm mục đích có thể mô tả nhiều trường thông tin của đối tượng bạn cần lưu

Cú pháp khai báo :

struct ten\_struct{

data\_type1 data\_field1;

data\_type2 data\_field2;

....

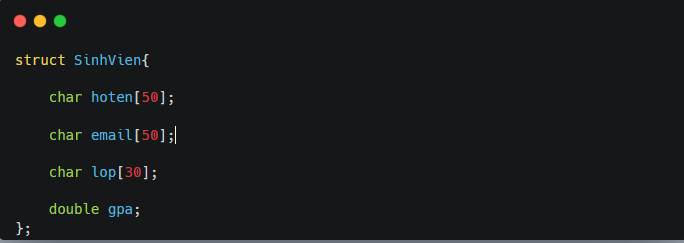
data\_typen data\_fieldn;};

Trong struct bạn sẽ liệt kê các trường thông tin mà bạn cần lưu cho cấu trúc bạn đang xây dựng kèm theo kiểu dữ liệu của trường thông tin đó

Ví dụ : Bạn muốn xây dựng một cấu trúc lưu sinh viên gồm :

* Họ tên - Chuỗi ký tự không quá 50
* Email - Chuỗi ký tự không quá 50
* Tên lớp - Chuỗi ký tự không quá 30
* Điểm gpa - Số thực

Ta sẽ xây dựng cấu trúc sinh viên như sau :

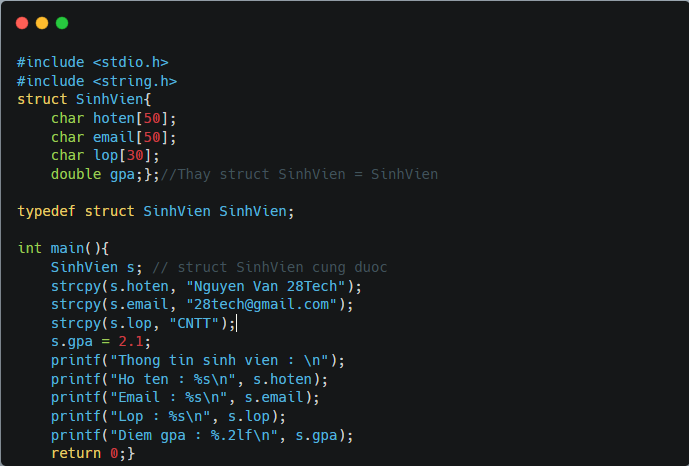


Sau khi xây dựng xong cấu trúc thì bạn có thể dùng nó như kiểu dữ liệu thông thường và khai báo các biến thuộc kiểu cấu trúc này.

Lưu ý là struct trong C khi khai báo bạn cần thêm từ khóa struct ở trước, nếu bạn không muốn khai báo từ khóa struct mỗi khi dùng cấu trúc thì có thể typedef để định nghĩa lại kiểu cho cấu trúc.

Để truy cập vào các trường dữ liệu của cấu trúc bạn dùng toán tử '.' (dot operator)

Ví dụ :



Output :

Thong tin sinh vien :

Ho ten : Nguyen Van 28Tech

Email : 28tech@gmail.com

Lop : CNTT

Diem gpa : 2.10

Bạn có thể gán 2 cấu trúc cho nhau, khi đó các trường thông tin của 2 biến struct là giống nhau

****\* Struct lồng nhau****

Struct có thể làm kiểu dữ liệu cho một trường thông tin trong một struct khác, khi đó ta có struct trong struct

Ví dụ bạn cần quản lý thông tin của 1 cuốn sách, cuốn sách này có tác giả cần lưu thông tin về tên và quốc tịch của tác giả đó

Khi đó bạn hãy xây dựng trước cấu trúc lưu tác giả, sau đó xây dựng struct sách. Trong struct sách sẽ có 1 biến thuộc kiểu tác giả để lưu thông tin

Ví dụ :



Output :

Thong tin sach :

Tieu de : Hanh trinh vo dich WC 2022

Ten tac gia : Lionel Messi

Quoc tich tac gia : Argentina

Gia ban : 500000VND

****\* Struct làm tham số cho hàm****

Sau khi xây dựng xong struct thì nó cũng là một kiểu dữ liệu vì thế mà bạn có thể viết hàm có tham số là struct hoặc trả về struct.

Trong ví dụ dưới đây mình sẽ viết hàm trả về 1 struct được nhập từ bàn phím và 1 hàm in ra thông tin của 1 struct ra màn hình.

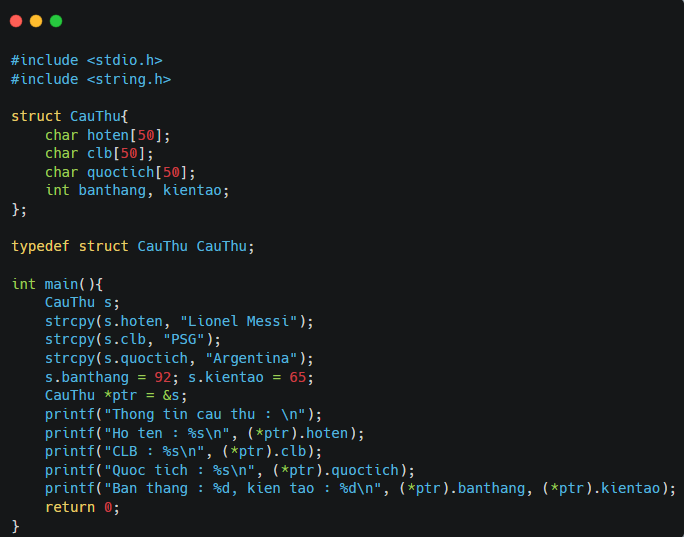
Ví dụ :



1. ****Con trỏ kiểu cấu trúc****

Con trỏ kiểu cấu trúc có thể sử dụng để trỏ tới biến cấu trúc, để truy cập các trường dữ liệu của biến cấu trúc thông qua con trỏ bạn có thể kết hợp toán tử giải tham chiếu

Ví dụ 1: Truy cập các trường dữ liệu của struct thông qua con trỏ.

Output :

Thong tin cau thu :

Ho ten : Lionel Messi

CLB : PSG

Quoc tich : Argentina

Ban thang : 92, kien tao : 65

Bạn có thể thay thế toán tử giải tham chiếu và toán tử '.' thành toán tử mũi tên '->'

Ví dụ 2 :

Output :

Thong tin cau thu :

Ho ten : Lionel Messi

CLB : PSG

Quoc tich : Argentina

Ban thang : 92, kien tao : 65

****\* Hàm nhập cấu trúc bằng con trỏ****

Để viết 1 hàm nhập dữ liệu cho biến cấu trúc từ bàn phím bạn cần dùng con trỏ để sau khi hàm kết thúc thì những thông tin bạn nhập sẽ được giữ nguyên.

Hàm này cần tham số là một con trỏ kiểu cấu trúc, nếu bạn không muốn nhập bằng cấu trúc thì có thể nhập thông tin cho 1 struct rồi trả về struct đó.

Ví dụ :

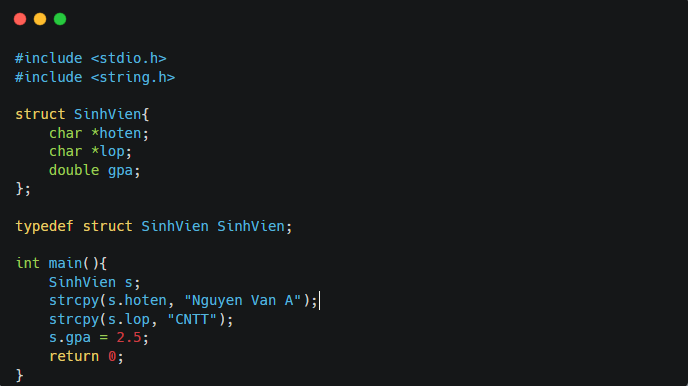


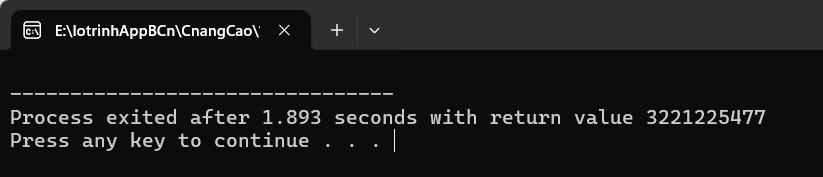
****\* Struct với trường dữ liệu là con trỏ****

Thông thường các trường dữ liệu của cấu trúc mà cần lưu mảng, ví dụ như là chuỗi ký tự thì ta hay dùng mảng tuy nhiên trường dữ liệu này cũng có thể là con trỏ.

Trong trường hợp các trường dữ liệu này là con trỏ thì khi bạn khai báo biến cấu trúc thì bản thân các trường dữ liệu này sẽ chưa có vùng nhớ để lưu, nên các bạn cần cấp phát động hoặc chỉ định vùng nhớ cho nó nếu không sẽ gây lỗi bộ nhớ.

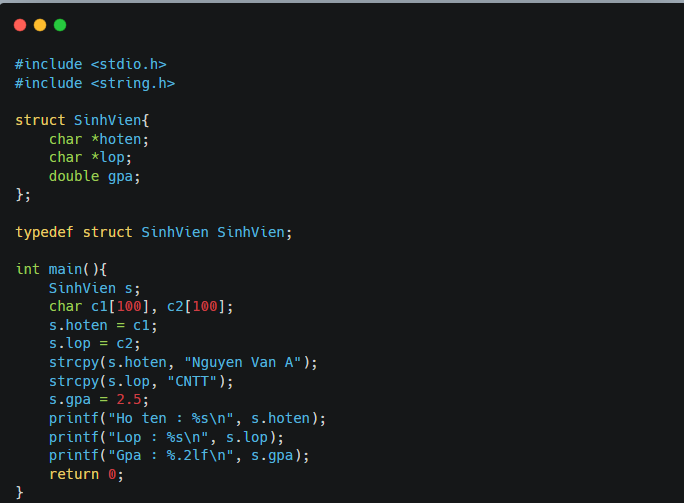
Ví dụ sau sẽ gây lỗi khi bạn cố tình lưu thông tin cho trường dữ liệu họ tên và lớp ở kiểu con trỏ :





Để không bị lỗi bộ nhớ thì bạn cần có vùng nhớ cho thuộc tính kiểu con trỏ này, có thể dùng một vùng nhớ tạm để lưu hoặc cấp phát động

Code dưới đây mình dùng 2 vùng nhớ tạm và cho 2 thuộc tính con trỏ tới tới vùng nhớ này



Output :

Ho ten : Nguyen Van A

Lop : CNTT

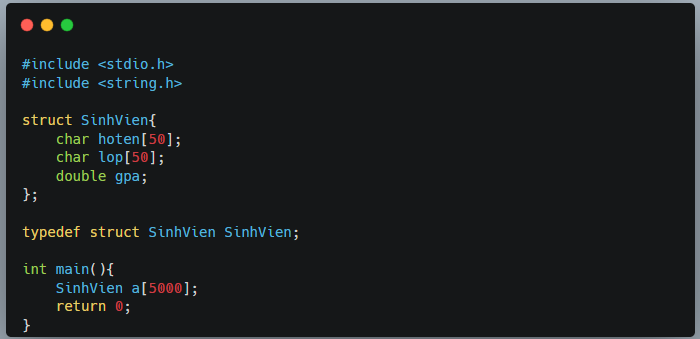
Gpa : 2.50

1. **Mảng cấu trúc**

Giả sử bạn cần quản lý danh sách khoảng 5000 sinh viên, khi đó thứ bạn cần đó là một mảng kiểu sinh viên có 5000 phần tử, mỗi phần tử trong mảng sẽ là một sinh viên

Sau khi có danh sách sinh viên được lưu trong mảng rồi thì bạn có thể sắp xếp, tìm kiếm, xóa, thay đổi thông tin …

Ví dụ : Khai báo mảng sinh viên có 5000 phần tử

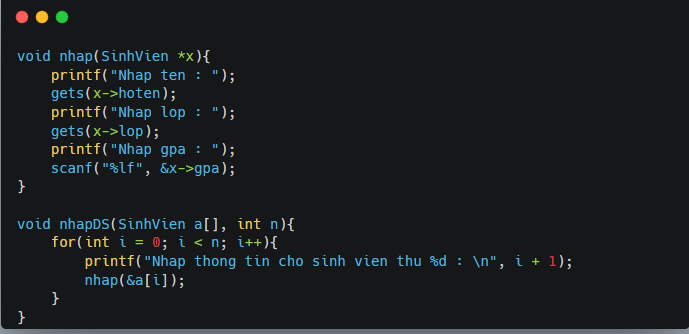


**\* **Hàm nhập mảng cấu trúc****

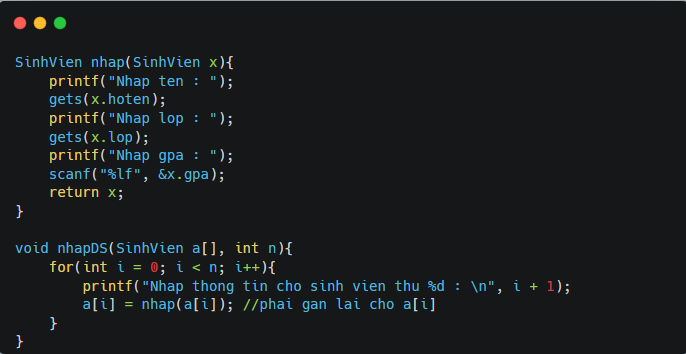
Để nhập thông tin cho 1 mảng cấu trúc, ví dụ danh sách N sinh viên thì bạn nên viết 1 hàm nhập thông tin cho 1 sinh viên và một hàm nhập danh sách sẽ gọi hàm nhập 1 sinh viên để nhập toàn bộ danh sách

Mình hướng dẫn 2 cách xây dựng hàm nhập thông tin cho 1 sinh viên, bạn có thể lựa chọn cách mình thấy dễ tiếp cận.

Ví dụ 1 : Hàm nhập sử dụng con trỏ



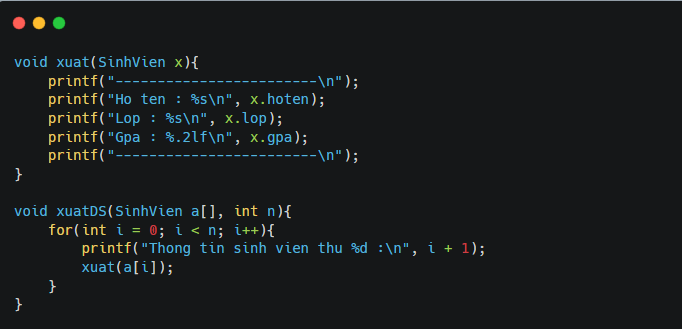
Ví dụ 2 : Hàm nhập không sử dụng con trỏ nhưng trả về kiểu cấu trúc



****\* Hàm xuất mảng cấu trúc****

Tương tự như để xuất mảng cấu trúc bạn nên xây dựng hàm để in thông tin cho 1 biến struct rồi xây dựng thêm 1 hàm in ra toàn bộ mảng

Ví dụ :



\* Sắp xếp mảng cấu trúc

Khi làm việc với mảng cấu trúc bạn cần thành thạo thao tác sắp xếp, ví dụ bạn muốn sắp xếp sinh viên theo tên, theo gpa hay vừa sắp xếp theo gpa vừa sắp xếp theo tên.

Trong các ví dụ ở bài viết này mình sẽ sử dụng cấu trúc sinh viên để demo

struct SinhVien{

char hoten[50];

char lop[50];

double gpa;};

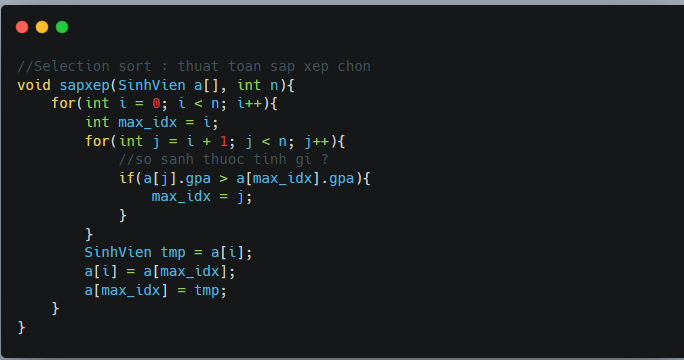
Muốn làm được các bài tập sắp xếp mảng cấu trúc bạn cần biết một vài thuật toán sắp xếp cơ bản hoặc nếu bạn được sử dụng hàm sắp xếp thì có thể sử dụng luôn qsort.

****Sắp xếp mảng cấu trúc bằng hàm tự xây dựng****

Thực ra thì sắp xếp dãy số hay sắp xếp mảng cấu trúc không có gì khác nhau về thuật toán, khi sắp xếp bạn cần lấy ra tiêu chí sắp xếp để so sánh giữa các phần tử trong mảng với nhau.

Ví dụ bạn muốn sắp xếp theo gpa thì khi so sánh 2 đối tượng trong mảng thì bạn lấy gpa ra để so sánh

Ví dụ  : Hàm sắp xếp sinh viên theo gpa giảm dần



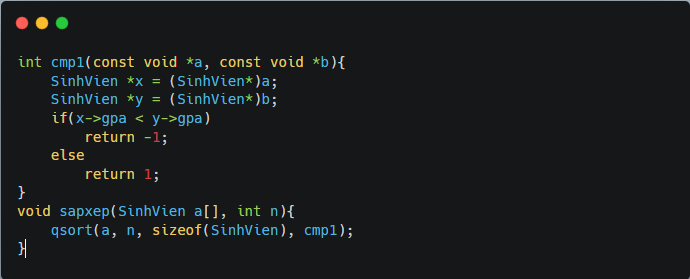
****Sắp xếp mảng cấu trúc bằng hàm qsort()****

Cách này sẽ tối ưu hơn so với cách bạn tự code, nếu bạn muốn tự code mà vẫn tối ưu như qsort() thì bạn phải sử dụng các thuật toán sắp xếp có độ phức tạp tốt như : Quicksort, merge sort, heapsort...

Nếu bạn sử dụng các thuật toán sắp xếp như chọn, chèn, đổi chỗ trực tiếp hay nổi bọt thì đều không tối ưu bằng cách dùng hàm này.

Nếu bạn chưa biết sử dụng hàm sắp xếp qsort() trong C thì có thể tham khảo [tại đây](https://blog.28tech.com.vn/c-ham-qsort" \t "https://blog.28tech.com.vn/_blank)

Ví dụ  : Hàm sắp xếp sinh viên theo gpa giảm dần



1. Union

Union cũng là kiểu dữ liệu mà người dùng tự định nghĩa tương tự như struct, union cũng có các thành phần dữ liệu để mô tả thông tin của đối tượng mà bạn đang muốn lưu trữ

Tuy nhiên điểm khác biệt với struct đó là struct thì các thành phần của nó có thể lưu trữ giá trị đồng thời tại cùng một thời điểm còn union thì ngược lại, tại mỗi thời điểm bạn chỉ có một thuộc tính của union lưu trữ 1 giá trị nào đó.

Tất cả các thành phần của union sẽ chia sẻ chung 1 vùng nhớ

Cú pháp :

union union\_name{

data\_type1 member1;

data\_type2 member2;

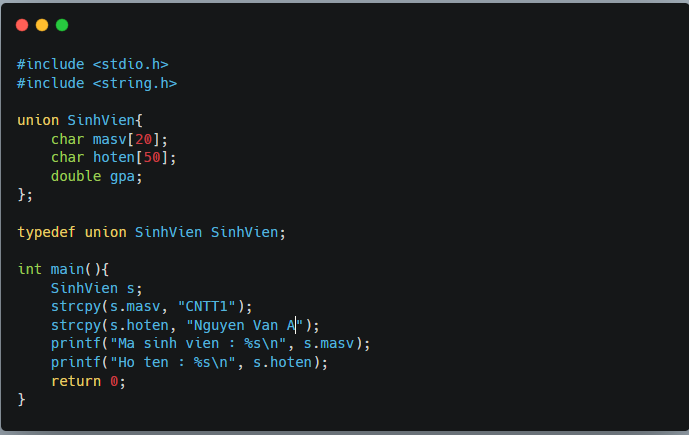
data\_type3 member3;

....

data\_typeN memberN;};

Để truy cập các thành phần của union bạn cũng dùng toán tử '.' với biến union bình thường hoặc toán tử '->' với kiểu con trỏ union

Ví dụ 1 : Các thuộc tính của union sẽ có giá trị giống nhau tại cùng 1 thời điểm



Output :

Ma sinh vien : Nguyen Van A

Ho ten : Nguyen Van A

Ví dụ 2 : Kích thước của union sẽ là kích thước của thuộc tính lớn nhất của nó



Output :

8

20

****Sự khác nhau giữa union và struct****

Union và struct giống nhau bao gồm :

* Đều là kiểu dữ liệu mà người dùng tự định nghĩa ra nhằm phục vụ các bài toán thực tế
* Có các thuộc tính thành phần
* Truy cập thuộc tính thông qua toán tử '.' hoặc toán tử '->' đối với biến kiểu con trỏ

Sự khác nhau giữa 2 kiểu dữ liệu người dùng định nghĩa được liệt kê dưới đây :

|  |  |
| --- | --- |
| ****struct**** | union |
| Kích thước của struct thường lớn hơn hoặc bằng tổng tất cả các kích thước của các thuộc tính thành phần | Kích thước của union bằng với kích thước của thuộc tính có kích thước lớn nhất của nó |
| struct có thể lưu trữ đồng thời nhiều giá trị cho các thuộc tính thành phần ở cùng một thời điểm | Chỉ có 1 thuộc tính thành phần của union có thể lưu giá trị tại 1 thời điểm |
| Khai báo với từ khóa struct | Khai báo với từ khóa union |