# File

**1. Tạo file**

Để làm việc với file thì đầu tiên bạn cần mở file đó lên, trong C để chương trình có thể làm việc với file bạn cần thông qua 1 con trỏ kiểu FILE

Cú pháp khai báo con trỏ FILE :  FILE \*f;

Sau khi khai báo con trỏ file thì bạn có thể dùng nó để mở file với hàm fopen(), hàm này nằm trong thư viện

Cú pháp :

FILE \*f;

f = fopen ("ten\_file", "mode");

Tham số thứ nhất trong fopen() là tên file mà bạn muốn mở, nếu bạn sử dụng file nằm trong cùng thư mục với file mã nguồn thì bạn chỉ cần tên file

Ví dụ :

#include <stdio.h>

int main(){

FILE \*f;

f = fopen("input.txt", "r");}

Nếu bạn muốn sử dụng file nằm đâu đó ở các thư mục khác với file mã nguồn thì bạn cần truyền vào tên file là đường dẫn duyệt đối.

Ví dụ :

#include <stdio.h>

int main(){

FILE \*f;

f = fopen("C:/tailieuC/Cnangcao/input.txt", "r");}

Tham số thứ 2 trong hàm fopen() chính là mode mở file, nó sẽ chỉ định việc bạn mở file lên để đọc, ghi file và đang đọc ghi với file text hay file nhị phân

**Các mode mở file chính :**

|  |  |  |
| --- | --- | --- |
| **Mode** | **Ý nghĩa** | **Chú ý** |
| **r** | **Mở file để đọc** | Nếu file không tồn tại thì hàm fopen() trả về con trỏ NULL |
| **rb** | **Mở file để đọc theo kiểu file nhị phân** | Nếu file không tồn tại thì hàm fopen() trả về con trỏ NULL |
| **w** | **Mở file để ghi** | Nếu file đã tồn tại thì sẽ làm việc với file đó, nếu file chưa tồn tại sẽ tạo 1 file mới |
| **wb** | **Mở file để ghi theo kiểu file nhị phân** | Nếu file đã tồn tại thì sẽ làm việc với file đó, nếu file chưa tồn tại sẽ tạo 1 file mới |
| **a** | Mở file text lên để ghi tiếp vào cuối file mà không xóa nội dung cũ trong file | Mở file text lên để ghi tiếp vào cuối file mà không xóa nội dung cũ trong file |
| **ab** | Mở file nhị phân lên để ghi tiếp vào cuối file mà không xóa nội dung cũ trong file | Nếu file đã tồn tại thì sẽ làm việc với file đó, nếu file chưa tồn tại sẽ tạo 1 file mới |
| **r+** | Mở file để vừa đọc vừa ghi | Nếu file không tồn tại thì hàm fopen() trả về con trỏ NULL |
| **rb+** | Mở file để vừa đọc vừa ghi theo kiểu nhị phân | Nếu file không tồn tại thì hàm fopen() trả về con trỏ NULL |
| **w+** | Mở file để vừa đọc vừa ghi | Nếu file đã tồn tại thì sẽ làm việc với file đó, nếu file chưa tồn tại sẽ tạo 1 file mới |
| **wb+** | Mở file để vừa đọc vừa ghi theo kiểu nhị phân | Nếu file đã tồn tại thì sẽ làm việc với file đó, nếu file chưa tồn tại sẽ tạo 1 file mới |
| **a+** | Mở file lên vừa để đọc và ghi vào cuối file | Nếu file đã tồn tại thì sẽ làm việc với file đó, nếu file chưa tồn tại sẽ tạo 1 file mới |
| ab+ | Mở file để vừa đọc vừa ghi vào cuối file theo kiểu nhị phân | Nếu file đã tồn tại thì sẽ làm việc với file đó, nếu file chưa tồn tại sẽ tạo 1 file mới |

Sau khi đọc ghi xong file bạn nên đóng file lại bằng hàm fclose()

Cú pháp :

FILE \*f; fclose( f );

**2. Ghi nội dung vào file**

Để ghi nội dung vào file bạn làm tương tự như bạn in nội dung ra màn hình, các hàm ghi nội dung vào file thường có thêm chữ f ở trước.

Để ghi nội dung vào file thì bạn chỉ cần dùng hàm fprintf() là đủ.

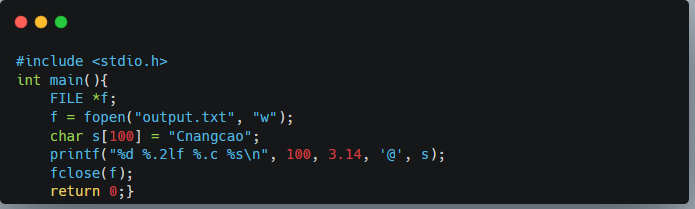
**Hàm fprintf()** :

Hàm fprintf() được dùng để ghi nội dung vào file tương tự như hàm printf

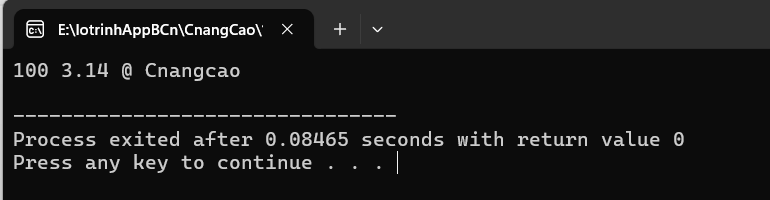
Cú pháp :

int fprintf ( FILE \* f, const char \* format, ... );

Ví dụ 1 : Ghi 1 số nguyên, 1 số thực, 1 kí tự và 1 chuỗi ký tự vào trong file output.txt



File output.txt



1. **Đọc nội dung từ file**

**\* File text :**

Để đọc nội dung từ file bạn sử dụng 2 hàm chính là fscanf() và fgets(), ngoài ra còn có fgetchar()

**Hàm fscanf()**

Hàm này tương tự hàm scanf() mà bạn hay dùng khi đọc từ bàn phím, bây giờ bạn đọc từ file.

Cú pháp :

int fscanf ( FILE \* f, const char \* format, ... );

Để đọc được file thì bạn cần biết file được lưu trữ như thế nào

Ví dụ 1 : Đọc số nguyên int, số thực double, chuỗi ký tự không có dấu cách trong file input.txt, sau đó in ra màn hình



File input.txt

28 3.14 Cnangcao

Output :

Data : 28 3.14 Cnangcao

**Hàm fgets()**

Hàm fgets() được sử dụng tương tự hàm gets(), dùng để đọc chuỗi ký tự có dấu cách.

Hàm này cũng bị trôi lệnh như hàm gets nên bạn cần lưu ý xử lý, ngoài ra nó còn đọc cả ký tự enter ở cuối dòng nên bạn cần loại bỏ ký tự enter này khỏi chuỗi

Cú pháp :

char \* fgets ( char \* str, int num, FILE \* f);

Tham số :

* char\* str : Xâu bạn muốn lưu nội dung đọc được
* num : Số lượng kí tự tối đa bạn muốn đọc
* f : Con trỏ file

Ví dụ :



File input.txt

28tech blog

Output :

Data : 28tech blog

**\* File nhị phân**

Hàm fread được sử dụng để **đọc dữ liệu nhị phân** từ một file vào bộ nhớ trong chương trình. Đây là hàm rất hiệu quả khi làm việc với file nhị phân, giúp đọc nhiều byte dữ liệu cùng một lúc.

### ****Cú pháp****

size\_t fread (void \*ptr, size\_t size, **size\_t** nmemb, FILE \*stream)

ptr:

* Con trỏ đến bộ nhớ nơi dữ liệu được lưu sau khi đọc từ file.
* Thường là địa chỉ của một biến hoặc một mảng.

size:

* Kích thước của mỗi phần tử cần đọc (tính bằng byte).
* Thường sử dụng sizeof() để xác định.

nmemb:

* Số lượng phần tử cần đọc.
* Nếu cần đọc toàn bộ file, bạn phải biết tổng số phần tử hoặc thực hiện đọc nhiều lần.

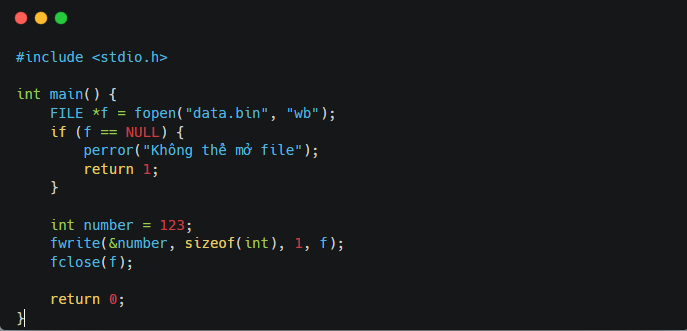
stream:

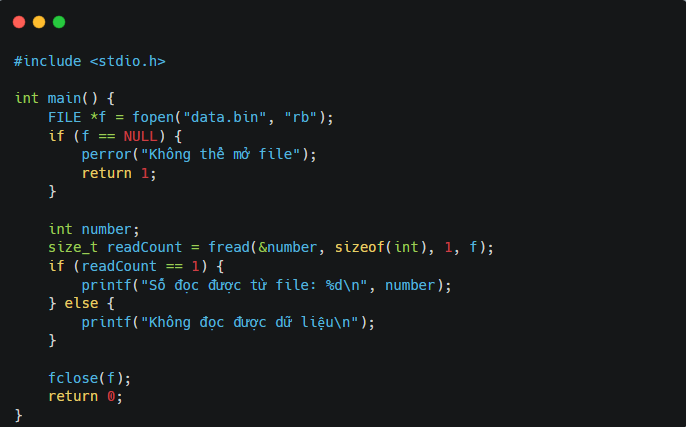
* Con trỏ file đã được mở bằng hàm fopen() với chế độ nhị phân ("rb" hoặc "r+b").

#### ****Giá trị trả về****

* Trả về số lượng phần tử thực sự đã đọc được.
* Nếu giá trị trả về nhỏ hơn nmemb, có thể file đã hết hoặc có lỗi trong quá trình đọc.

**Ví dụ :** Tạo file data.bin trước (chứa số 123):





**Kết quả:**

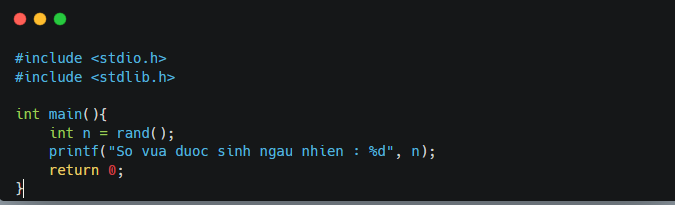
Số đọc được từ file: 123

**\* Hàm Sinh Số Ngẫu Nhiên**

Rand () :

Hàm rand() trong C được sử dụng để sinh số ngẫu nhiên 1 số nguyên trong khoảng từ 0 tới RAND\_MAX, trong đó RAND\_MAX là một hằng số có giá trị là 32767. Giá trị này có thể khác ở một số chuẩn C khác nhau.

Ví dụ:



Output :

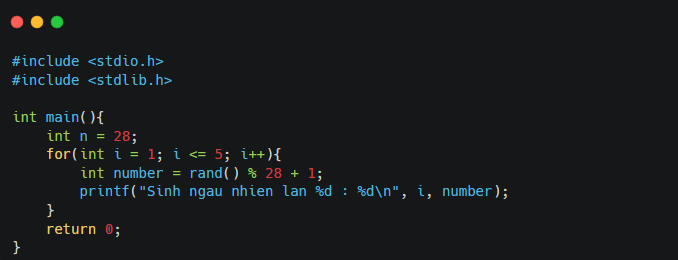
So vua duoc sinh ngau nhien : 41

Trên máy tính của bạn có thể kết quả sẽ ra khác 0 tời 32766

Ví dụ 2 : Sinh số ngẫu nhiên từ 1 tới N

Cú pháp : rand() % N + 1

Đối với bài toán này bạn vẫn sử dụng hàm rand() nhưng cần chia dư số do hàm rand() trả về với N, vì khi bạn chia dư cho N thì số dư sẽ chạy từ 0 tới N - 1, bạn cộng thêm 1 vào số dư sẽ có thể sinh được các số từ 1 tới N



Output :

Sinh ngau nhien lan 1 : 14

Sinh ngau nhien lan 2 : 16

Sinh ngau nhien lan 3 : 7

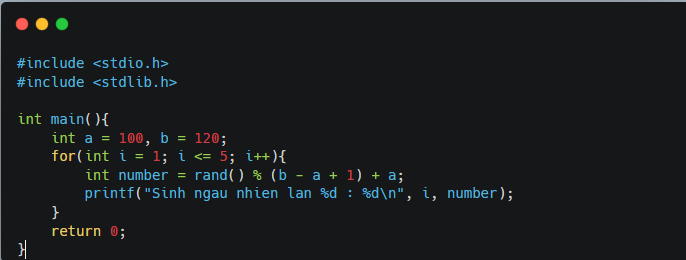
Sinh ngau nhien lan 4 : 13

Sinh ngau nhien lan 5 : 18

Ví dụ 3 : Sinh số ngẫu nhiên từ a đến b

Cú pháp : rand() % (b - a + 1) + a

Sinh số từ 100 đến 120



Output :

Sinh ngau nhien lan 1 : 120

Sinh ngau nhien lan 2 : 108

Sinh ngau nhien lan 3 : 113

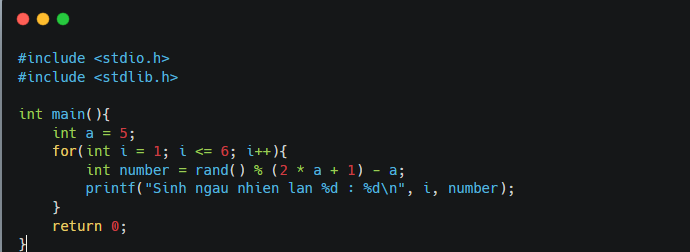
Sinh ngau nhien lan 4 : 119

Sinh ngau nhien lan 5 : 117

Ví dụ 4 : Sinh số ngẫu nhiên từ -a tới a

Cú pháp : rand() % (2a + 1) - a

Ví dụ khi a = 5, bạn cần sinh ra các số 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 và trừ đi a khi đó các giá trị ngẫu nhiên thu được sẽ chạy từ -5 tới 5



Output :

Sinh ngau nhien lan 1 : 3

Sinh ngau nhien lan 2 : 4

Sinh ngau nhien lan 3 : 4

Sinh ngau nhien lan 4 : -4

Sinh ngau nhien lan 5 : 2

Sinh ngau nhien lan 6 : 0

Srand()

Hàm srand() trong C được sử dụng để tạo seed - tạm dịch là hạt giống cho việc sinh ngẫu nhiên. Trong các ví dụ ở trên nếu bạn chạy chương trình của mình nhiều lần thì kết quả sinh ngẫu nhiên ở các lần sẽ giống nhau.

Để tránh trường hợp này thì bạn nên sử dụng thêm hàm srand() để tạo seed cho hàm rand() trước khi sinh ngẫu nhiên

Cú pháp :

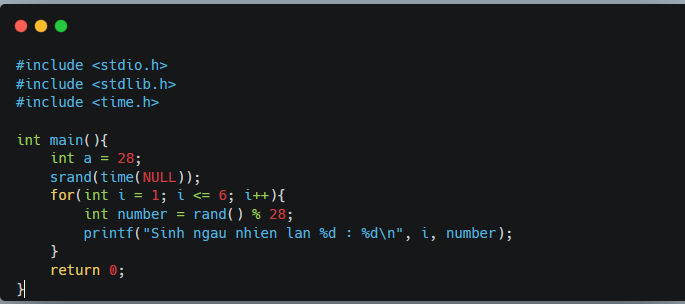
void srand (unsigned int seed);

Nếu tham số seed của srand() là 1 thì tương tự như việc bạn không sử dụng srand() trước hàm rand() như các ví dụ ở phần 1

Thông thường thì tham số seed này được lựa chọn dựa vào thời gian thông qua hàm time() trong thư viện time.h

Hàm time() với tham số con trỏ NULL trả về số giây tính từ 00:00 ngày 01/01/1970 tới thời gian hiện tại

Ví dụ :



Output :

Sinh ngau nhien lan 1 : 3

Sinh ngau nhien lan 2 : 9

Sinh ngau nhien lan 3 : 0

Sinh ngau nhien lan 4 : 21

Sinh ngau nhien lan 5 : 19

Sinh ngau nhien lan 6 : 22

**Mục đích của** srand(time(NULL)):

* Tạo ra một **hạt giống ngẫu nhiên** thay đổi theo thời gian.
* Nếu không dùng srand(time(NULL)) mà chỉ gọi rand(), chương trình sẽ luôn sinh ra **cùng một dãy số** ngẫu nhiên trong mỗi lần chạy (vì hạt giống mặc định là 1).

Sau khi sử dụng srand() trước khi sử dụng rand() thì mỗi lần chương trình bạn chạy sẽ sinh ra kết quả ngẫu nhiên khác nhau. Điều này là do mỗi lần chạy bạn đều sử dụng một seed khác nhau cho việc sinh số ngẫu nhiên.