# Hàm fopen(): mở một file dữ liệu

# Hàm fopen() là hàm có sẵn trong thư viện cstdio, vì vậy trước khi sử dụng nó các bạn nhớ khai báo thư viện đã nhé: #include<cstdio>.

# Cú pháp:

# FILE\* fopen (filename, mode);

# Trong đó:

* filename: tên file hoặc đường dẫn đến file đó

vd: /home/sv/cntt/lab7/test.c

* mode: chế độ truy cập file.

# Cụ thể:

# 

**Hàm trả về:**

* Nếu thành công, hàm fopen() trả về một chuỗi là địa chỉ của file đó
* Khi thất bại, nó trả về một chuỗi null.

Hàm fopen() có thể đọc các file nằm bất kì đường dẫn nào miễn là đường dẫn ấy phải tuyệt đối, rõ ràng và chi tiết

Ví dụ: Nếu file mã nguồn của chương trình là main.c và file cần đọc là input.txt và nó nằm trong đường dẫn /path/to/project/data/input.txt , để mở file input.txt trong chương trình, ta sử dụng đường dẫn tuyệt đối như sau:

FILE \*fp = fopen("/path/to/project/data/input.txt", "r");

1. **Hàm open()**

giống với fopen() nhưng dùng để mở một file đặc biệt gọi là fifo hay named pipe (đường ống) mà các tiến trình thường xuyên sử dụng để giao tiếp nhau bằng kỹ thuật đường ống pipe, và chỉ có 2 chế độ truy cập:

* O\_RDONLY (chỉ đọc) : tiến trình chỉ có thể đọc dữ liệu từ đường ống và không thể ghi dữ liệu vào đường ống.
* O\_WRONLY (chỉ ghi): tiến trình chỉ có thể ghi dữ liệu vào đường ống và không thể đọc dữ liệu từ đường ống.

 Minh họa:

Lưu ý: file đặc biệt này sẽ có kiểu int chứ không phải FILE\*

1. Hàm fprintf(): ghi dữ liệu vào file

Cú pháp:

**int fprintf (FILE \* file , “ format” , str1, str2, ...)**

**Tham số:**

* **stream**: con trỏ tới đối tượng **file dữ liệu**.
* **format**: chuỗi + đặc tả ghi vào file dữ liệu. vd: “ output: %d %d %d “
* **str1, str2**: các biến chứa dữ liệu được ghi vào file

**Cách hoạt động: Ghi các biến str1, str2 vào file**

**lưu ý: dùng khi file được mở ở chế độ “w” bằng fopen**



**Lưu ý:**

Hàm **fprintf()** trả về một số thuộc kiểu **int**, là số ký tự có trong chuỗi đã tạo nếu ghi file thành công.

Hàm **fprintf()** trả về giá trị **EOF** nếu ghi file thất bại.

1. Hàm fscanf(): Đọc dữ liệu một file được tạo ra từ fopen()

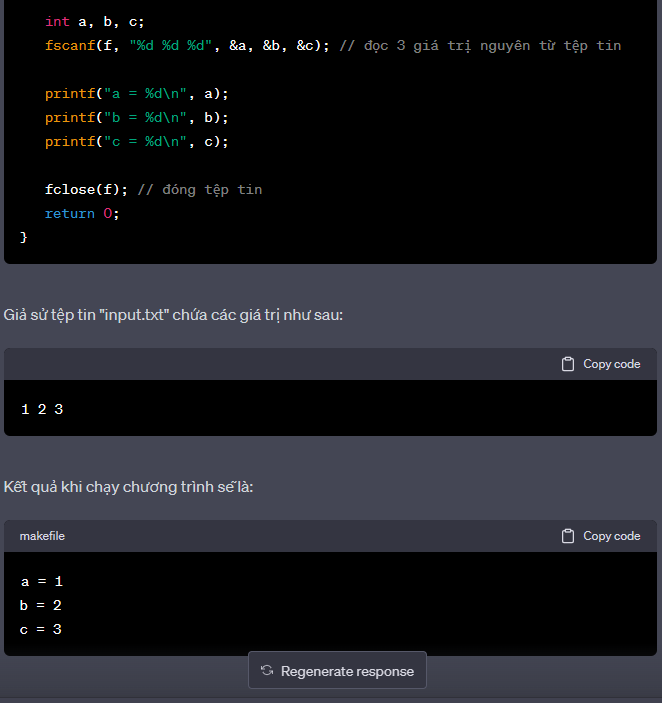
Cú pháp:

int fscanf(FILE\* file, “ format ” , str1, str2 …)

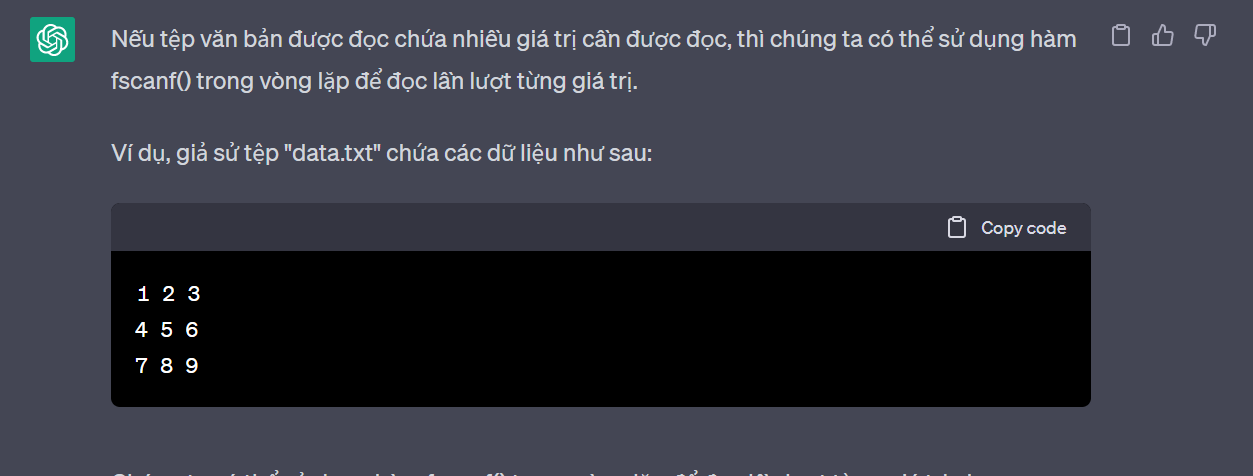
**Tham số**

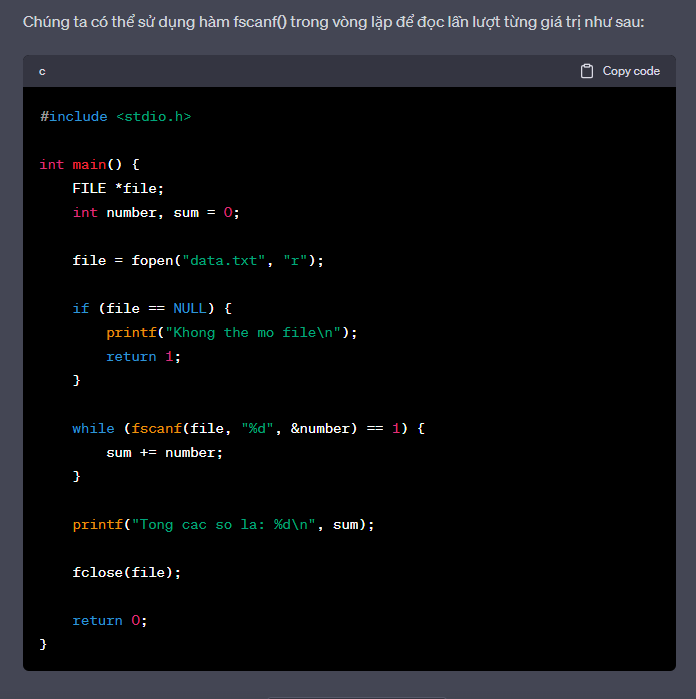
* **file** – con trỏ trỏ đến một file chứa dữ liệu (biến đặc biệt có kiểu dữ liệu FILE\*)
* **format** – đặc tả của các dữ liệu trong file đó
* str1, str2, … - các biến lưu trữ các dữ liệu trong file đó

Cách hoạt động: Đọc từng dữ liệu trong file rồi copy chúng vào các biến str1, str2,.. với format tương ứng.

Lưu ý: Nếu file được lấy dữ liệu không được mở bằng hàm fopen(), thì bạn không thể sử dụng fscanf() để đọc dữ liệu từ file đó.

Nếu đặt trong vòng lặp While, giá trị n sẽ thay đổi mỗi lần đọc từ file, riêng giá trị trả về của fscanf() luôn là 1.





## Hàm fgets() : lấy dữ liệu từ một file

Cú pháp :

fgets (char [] str, int n , FILE\* file)

**Tham số**

* **str** -- mảng char nơi dữ liệu được lưu trữ.
* **n** -- Đây là số ký tự tối đa được đọc (bao gồm ký tự null cuối cùng).
* **file** -- đường dẫn file nếu dữ liệu là lấy từ một file, hoặc là stdin nếu dữ liệu đầu vào do người dùng nhập từ bàn phím

**Cách hoạt động : lấy n byte dữ liệu từ file đưa vào mảng str**

**Trả về giá trị**

* Nếu thành công, hàm trả về cùng tham số str. Nếu bắt gặp EOF và không có ký tự nào đã được đọc, nội dung của str vẫn không thay đổi và một con trỏ null được trả về.
* Nếu một lỗi xuất hiện, một con trỏ null được trả về.

Minh họa:

#include <stdio.h>

int main()

{

FILE \*fp;

char str[60];

// lưu ý nhỏ: fget() chỉ có thể lấy dữ liệu từ file đã được mở từ fopen() , nếu

// không thì nó sẽ trả về NULL

/\* mo file de doc \*/

fp = fopen("baitapc.txt" , "r");

if(fp == NULL)

{

perror("Xay ra loi trong khi doc file");

return(-1);

}

if( fgets (str, 60, fp)!=NULL ) // lấy 60 byte dữ liệu từ file fp và đưa vào trong mảng str, nếu

{

/\* Ghi noi dung len stdout \*/

puts(str);

}

fclose(fp);

return(0);

}

1. Hàm memset(): đặt giá trị cho phần tử mảng

Cú Pháp:

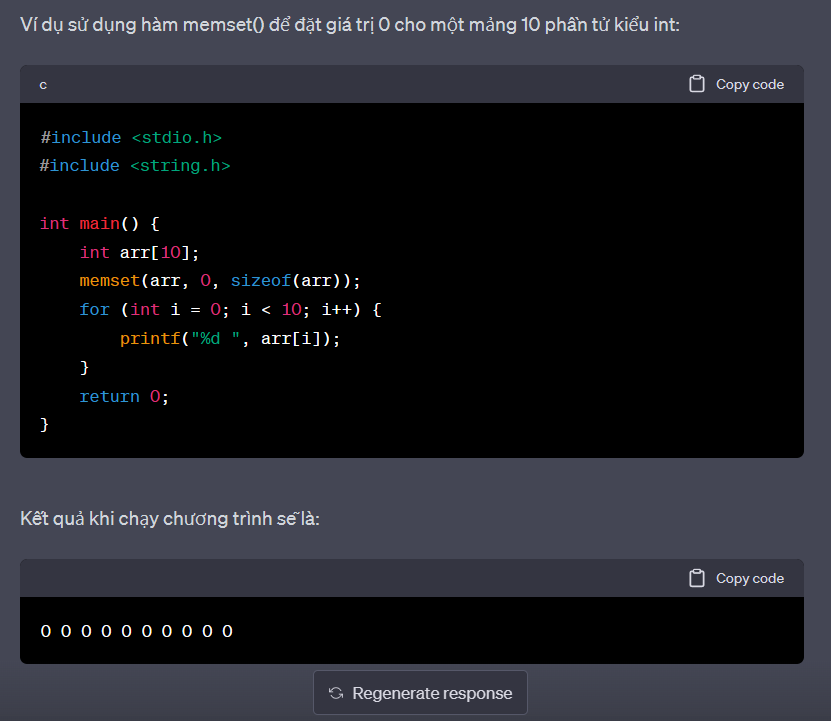
void \*memset(void \*ptr, int value, size\_t n);

Trong đó:

* ptr là con trỏ tới vị trí đầu tiên của vùng nhớ cần đặt giá trị. (mảng)
* value là giá trị cần thiết để đặt cho các phần tử trong vùng nhớ.
* n là số byte cần đặt giá trị.

Hàm này sẽ trả về con trỏ tới vùng nhớ ban đầu, tức là ptr.

Cách hoạt động: thay n phần trong mảng bằng giá trị value



1. Hàm read(): đọc dữ liệu một file descriptor

Cú pháp:

ssize\_t read(int fd, void\* buf, size\_t count);

Trong đó:

* fd là file descriptor của file cần đọc dữ liệu.
* buf là con trỏ tới vùng nhớ cần lưu trữ dữ liệu đọc được (hoặc là mảng)
* count là số byte dữ liệu tối đa cần đọc.

Cách hoạt động: đọc dữ liệu từ một file descriptor (một số nguyên đại diện cho một tập tin, ổ đĩa, hoặc thiết bị khác), và lưu trữ dữ liệu đọc vào một mảng buf

Hàm read sẽ trả về số byte dữ liệu thực sự đã đọc được (hoặc 0 nếu đã đọc đến cuối file), hoặc -1 nếu có lỗi xảy ra.

# Hàm pipe(): tạo đường ống cho 2 tiến trình cha – con (unnamed pipe)

# 

# Hàm mkfifo(): tương tự pipe nhưng dành cho 2 tiến trình không phải cha con (named piped)