

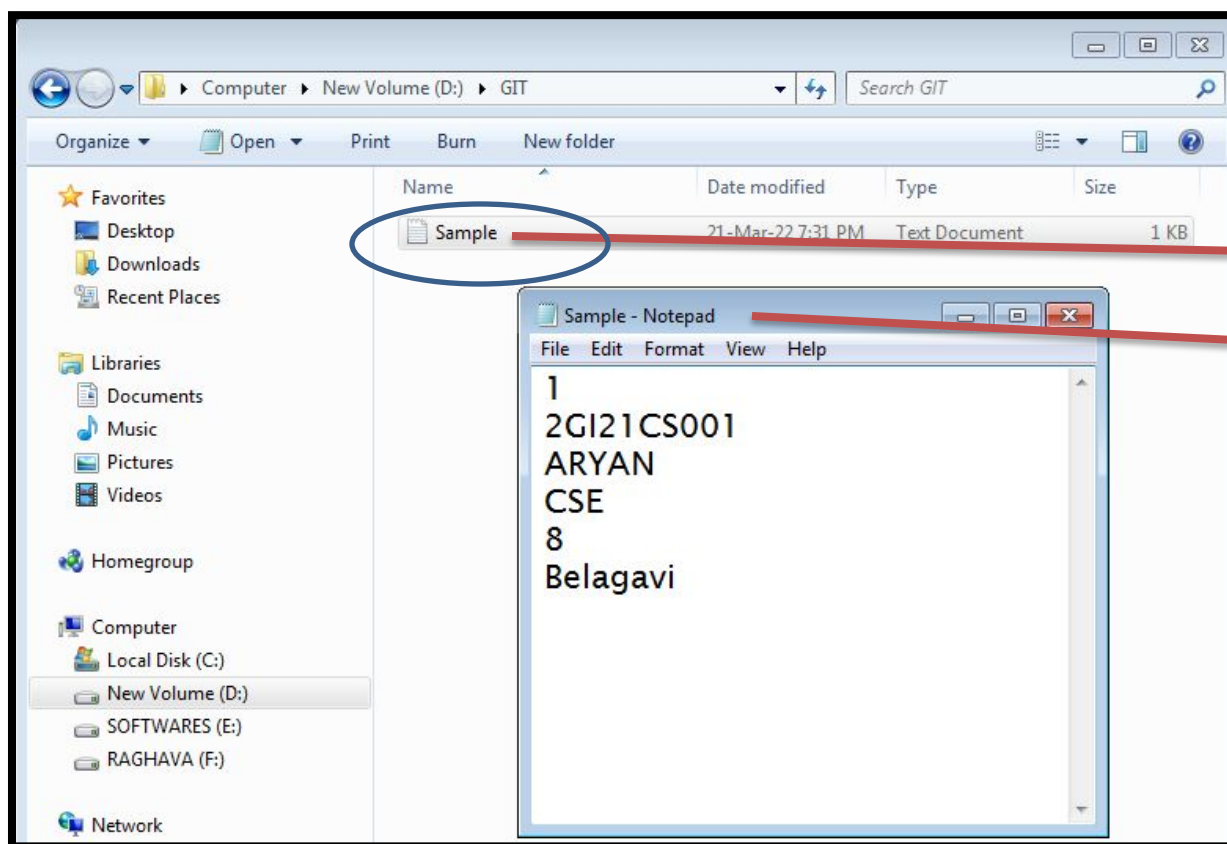
# Files in C

# Why Files?

- ❖ Till now, we were dealing with data associated with the programs in the console/terminal with the help of console oriented I/O functions such as `printf()` and `scanf()` functions.
- ❖ This data can be an input to the program or it can be an output of the program.
- ❖ Handling data in the console becomes difficult as the size of the data increases because in console oriented I/O operations the data is lost as soon as the console/terminal is closed or the program execution is completed or the computer is turned off.
- ❖ Therefore, in order to retain such data instead of losing it in the terminal a FILE can be used.

# What is a File?

- ❖ A file is a place on the disk where a group of related data is stored.
- ❖ A file is a collection of data stored on a secondary storage device such as hard disk.



An example of a text file.  
Every file has some name and an extension. Here the name of the file is Sample and extension is .txt

# Types of Files



## Binary files

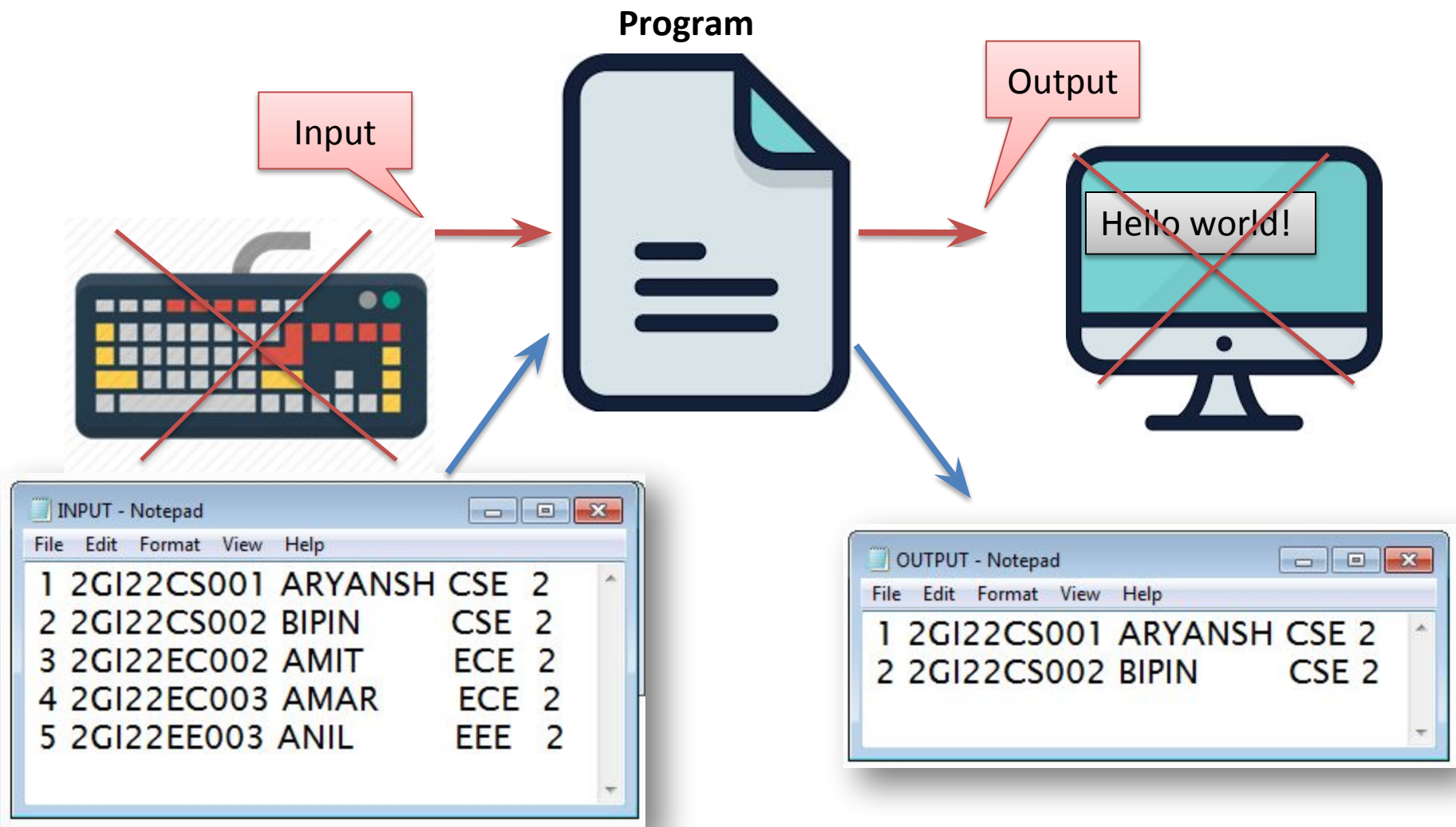


## Text files

- ❖ Binary files are mostly the **Executable files, compiled programs, graphic (image) files, audio/video** files in your computer.
- ❖ Instead of storing the data in plain text, they store data in the binary form (0's and 1's).
- ❖ Appropriate software/programs are required to interpret the data present in these files.

- ❖ Text files are the normal **.txt** files. You can easily create text files using any simple text editors such as Notepad.
- ❖ When you open those files, you'll see all the contents within the file as plain text. You can easily edit or delete the contents.

Instead of giving input to the program through console I/O operations, store all data into a file and design a C program to read this data as input. Similarly instead of displaying the output data on the terminal design a program to write the output of the program into a file. This helps to store the data permanently in the computers secondary memory and can be referred whenever required.



# Using Files in C

To use files in C, we must use the following steps,

1. **Declare** a File pointer variable
2. **Open** the file (in any one of the modes- ***r w a***)
3. **Process** the file (**read/write**)
4. **Close** the file

# 1. Declare a File pointer variable

- ❖ In order to perform any operations on the file we must first have an access to the files location(address).
- ❖ This is done by creating a pointer pointing to the **FILE structure** (defined in stdio.h). This is accomplished by declaring a FILE pointer.

## Syntax for declaring a FILE pointer:

```
FILE *file_pointer_name;
```

An Example,

```
FILE *fp;
```

## 2. Opening the file

- ❖ A file must be opened before data can be read from or written to it.
- ❖ In order to open a file and associate it with a stream, the **fopen()** function is used.
- ❖ fopen() function returns a pointer to the structure on successful opening of the file and if it fails it returns NULL.

### Syntax for opening a file:

```
FILE *fp;  
  
fp  
  
fopen("filename", "mode");
```

An example,

**r** for reading  
**w** for writing  
**a** for appending

```
FILE *fp;  
fp = fopen("sample.txt", "w");
```



## 2. Opening the file


- ❖ If the file is in the current working directory, then just name of the file is enough while opening it.

**Example:** `fopen("sample.txt", "r");`

- ❖ However, full path of the files location should be given if it is not in the current working directory.

**Example:** `fopen("D:\\Programs\\sample.txt", "r");`

- ❖ As `fopen()` returns `NULL` if the file does not exist, it's the duty of programmer to always check if the file exists or no before proceeding further.

An Example: 

```
#include<stdio.h>
void main()
{
    FILE *fp;
    fp = fopen("student.txt", "r");
    if(fp == NULL)
        printf("File doesn't exists");
    else
        printf("File exists");
    fclose(fp);
}
```

## Different modes for opening a file

Mode	Description	
<b>r</b>	Opens the file for reading.	If the file does not exist, fopen() returns NULL.
<b>w</b>	Opens the file for writing.	If the file exists, its contents are overwritten. If the file does not exist, a file will be created.
<b>a</b>	Open for append. Data is added to the end of the file.	If the file does not exist, a new file will be created.
<b>r+</b>	Open for both reading and writing.	If the file does not exist, fopen() returns NULL.
<b>w+</b>	Open for both reading and writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
<b>a+</b>	Open for both reading and appending.	If the file does not exist, it will be created.

### 3. Processing the file (**reading/writing**)

There are various functions available for reading data from the file and writing data to the file.

Functions for **reading** data from a file:

- ❖ fgetc()
- ❖ fgets()
- ❖ fscanf()
- ❖ fread()

Functions for **writing** data to a file:

- ❖ fputc()
- ❖ fputs()
- ❖ fprintf()
- ❖ fwrite()

## 4. Close the file

- ❖ To close the opened file, the `fclose()` function is used which disconnects a file pointer from a file
- ❖ After `fclose()` function has disconnected the file pointer from the file, the pointer can be used to access a different file.
- ❖ Along with closing the file it also flushes all the buffers that are maintained for that file.

### Syntax:

```
fclose(FILE *fp);
```

```
Example, fclose(fp);
```

## Reading data from a file using fgetc():

- ❖ fgetc() is used to read one character at a time from a file.
- ❖ fgetc() returns the character as an int or return EOF to indicate an error or end of the file.

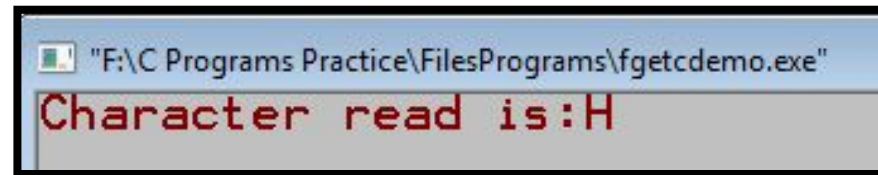
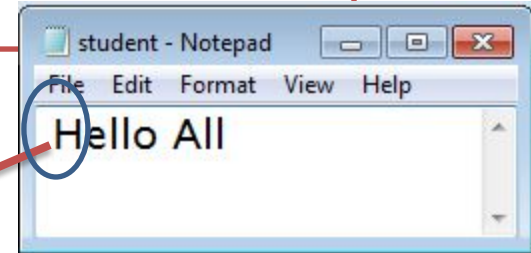
### Syntax:

```
int fgetc(FILE *stream);
```

### Example:

```
FILE *fp;  
char ch;  
fp = fopen("student.txt", "r");  
ch = fgetc(fp);
```

H is read from  
student.txt file  
and stored in ch



```
#include<stdio.h>  
void main()  
{  
    FILE *fp;  
    char ch;  
    fp = fopen("student.txt", "r");  
    ch = fgetc(fp);  
    printf("Character read is:%c\n", ch);  
    fclose(fp);  
}
```

## Writing data to a file using fputc():

- ❖ fputc() is used to write one character(a byte) to the file.
- ❖ fputc() returns the character value that it has written on success or EOF in case of error.

### Syntax:

```
int fputc(int c, FILE *stream);
```

### Example:

```
FILE *fp;  
char ch= 'A';  
fp = fopen("student.txt", "w");  
fputc(ch, fp);
```

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
FILE *fp;
```

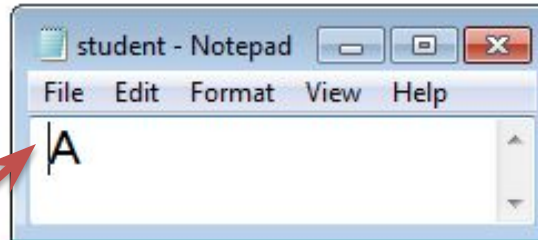
```
char ch = 'A';
```

```
fp = fopen("student.txt", "w");
```

```
fputc(ch, fp);
```

```
fclose(fp);
```

```
}
```



## Reading data from a file using fgets():

- ❖ fgets() function reads at most one less than the number of characters specified by size from the given stream.
- ❖ fgets() terminates as soon as it encounters either a newline/EOF/or error.

### Syntax:

```
char fgets(char *s, int size, FILE  
*stream);
```

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
FILE *fp;
```

```
char s[10];
```

```
fp = fopen("student.txt", "r");
```

```
fgets(s, 9, fp);
```

```
printf("Read string:%s", s);
```

```
fclose(fp);
```

```
}
```

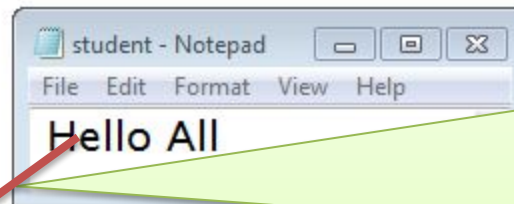
### Example:

```
FILE *fp;
```

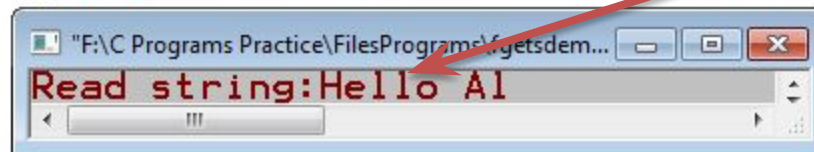
```
char s[10];
```

```
fp = fopen("student.txt", "r");
```

```
fgets(s, 10, fp);
```



First 9-1 Characters are read from the file student.txt and stored in string s, and with help of printf() s is printed and you can observe that only Hello Al is printed living the last 'l' because of size given as 9.



## Writing data to a file using fputs():

- ❖ fputs() function is used to write a line to a file.
- ❖ fputs() returns 0 on success or EOF in case of any error.

### Syntax:

```
int fputs(char *s, FILE *stream);
```

### Example:

```
FILE *fp;  
char s[10]= "Hello All";  
fp = fopen("student.txt", "w");  
fputs(s, fp);
```

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
FILE *fp;
```

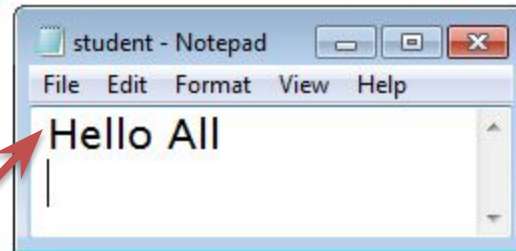
```
char s[10] = "Hello All";
```

```
fp = fopen("student.txt", "w");
```

```
fputs(s, fp);
```

```
fclose(fp);
```

```
}
```





## Reading data from a file using fscanf():

- ❖ fscanf() function is used to read formatted data from the file.
- ❖ fscanf() terminates as soon as it encounters either a newline/EOF/or error.

### Syntax:

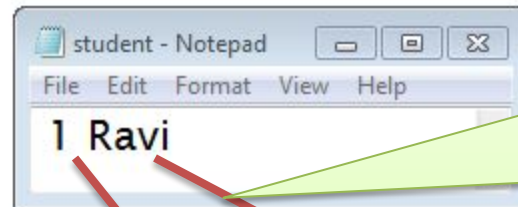
```
int fscanf(FILE *stream, "format specifiers", variables);
```

```
#include<stdio.h>
void main()
{
```

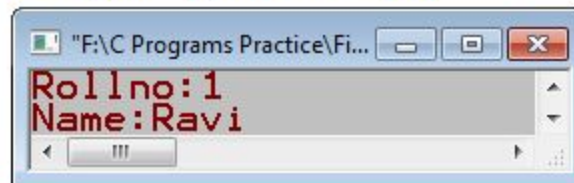
```
FILE *fp;
char name[10];
int rollno;
fp = fopen("student.txt", "r");
fscanf(fp, "%d%s", &rollno, name);
printf("Rollno:%d\n", rollno);
printf("Name:%s\n", name);
fclose(fp);
```

### Example:

```
fscanf(fp, "%d%s", &rollno, name);
```



1 and Ravi is read from student.txt file and stored in rollno and name variables respectively and are printed using printf().



## Writing data to a file using fprintf():

❖ fprintf() function is used to write formatted data to a file.

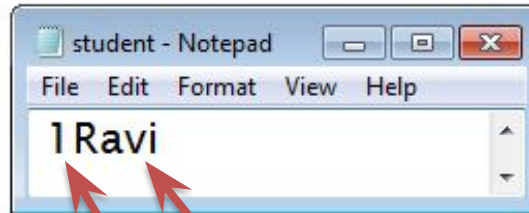
### Syntax:

```
int fprintf(FILE *stream, "format specifiers", variables);
```

### Example:

```
fprintf(fp, "%d%s", &rollno, name);
```

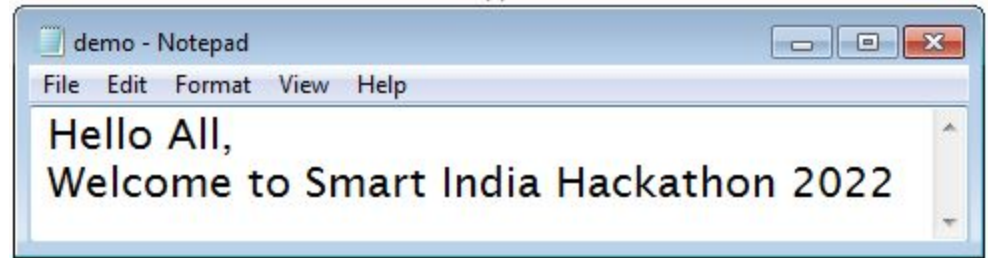
```
#include<stdio.h>
void main()
{
    FILE *fp;
    char name[10]="Ravi";
    int rollno=1;
    fp = fopen("student.txt", "w");
    fprintf(fp, "%d%s", rollno, name);
    fclose(fp);
}
```



The values of rollno and name variables are written to the file student.txt.

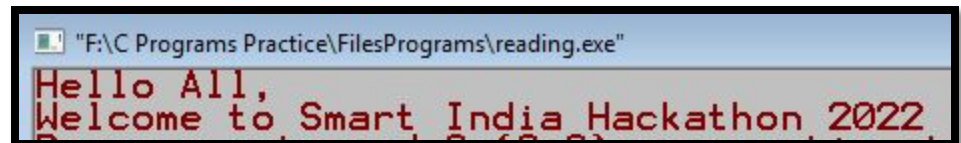
Write a C program to read and print all the data present in the demo.txt file.

```
#include<stdio.h>
void main()
{
    FILE *fp;
    char ch;
    fp = fopen("demo.txt", "r");
    ch = fgetc(fp);
    while (ch != EOF)
    {
        printf("%c", ch);
        ch = fgetc(fp);
    }
    fclose(fp);
}
```



As we know fgetc() reads one character at a time from the file wherever fp is pointing to. So a loop is used to repeatedly read the characters one after the other and printed using printf()

Output:



Write a C program to copy the contents of one file to another file.

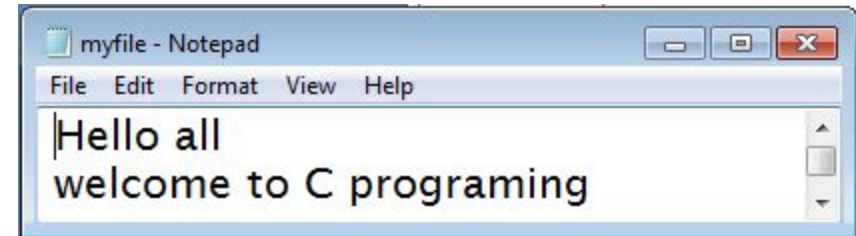
```
#include<stdio.h>
void main()
{
    FILE *fp1,*fp2;
    char ch;
    fp1 = fopen("source.txt","r");
    fp2 = fopen("destination.txt","w");
    while ((ch=fgetc(fp1)) != EOF)
    {
        fputc(ch,fp2);
    }
    fclose(fp1);
    fclose(fp2);
}
```





Write a C program to create a file named myfile and write content that the user types from the keyboard till the user enters 0.

```
#include<stdio.h>
void main()
{
    FILE *fp;
    char ch;
    fp = fopen("myfile.txt", "w");
    if(fp==NULL)
        printf("Error File doesn't exist\n");
    else
    {
        printf("File myfile.txt created\n");
        while((ch=getchar())!='0')
        {
            fputc(ch, fp);
        }
    }
    fclose(fp);
}
```



Output:



**Write a C program that creates a file reading contents that the user types from the keyboard till EOF. The text in this file must be in lowercase. There could be multiple blanks in between some words. Create another file in which the same content is copied in UPPERCASE and with only one blank in between the words that contained multiple blanks.**