

File Management in 'C'

Prepared By: Padmavathi Bindulal
Assistant Professor, CE,CSPIT.

Contents

Introduction,
Defining and Opening a file,
Closing a file,
Modes of file,
Read & write single character and integer to file,
Use of fprintf and fscanf functions.
Error handling functions,
Random access of files using ftell, rewind, fseek,
Command line argument.

Need of Files

Problems with Console I/O

- It becomes difficult and time consuming to handle large volumes of data through console.
- The entire data is lost when either the program is terminated or the computer is turned off.

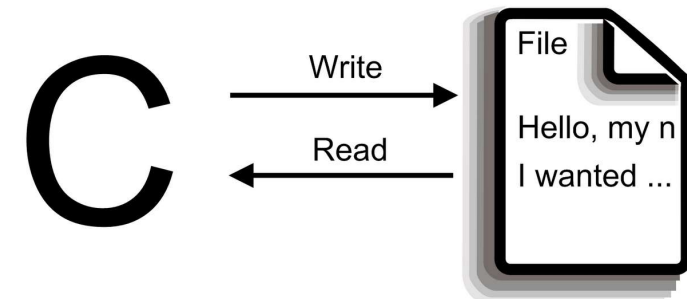
It is necessary to have a more flexible approach where data can be stored on the disks and read whenever necessary, without destroying the data.

This is achieved by using the concept of **Files**.

What is a File?

A file is a collection of related data stored on a disk. C supports a wide range of functions that have the ability to perform basic file operations, which include:

- Naming a file
- Opening a file
- Reading data from a file
- Writing data to a file
- Closing a file



2 methods to perform file operations in C

Low level-I/O : Uses Unix System calls

High level-I/O: C library provides various pre-defined functions for handling files.

Function	Operation
fopen()	Creates a new file / opens an existing file
fclose()	Closes a file which has been opened for use
getc()	Reads a character from the file
putc()	Writes a character to the file
fprintf()	Write data values to a file
fscanf()	Reads a set of data values from a file
getw()	Reads an integer from the file
putw()	Writes an integer to a file
fseek()	Sets the position to the desired point in the file
ftell()	Gives the current position in the file
rewind()	Sets the position to the beginning of the file

* Not all of them are supported by all the compilers

Defining and Opening a File

Requirements to
create/open a
file

File name

Data structure

Purpose

- String of characters
- Primary name and an optional period with extension
- Examples:
Input.data
Store
Prac.c
Text.out
- FILE is a defined type in the I/O library
- Specify what to do with file
- Eg: read, write, modify.

General format for declaring
and opening file

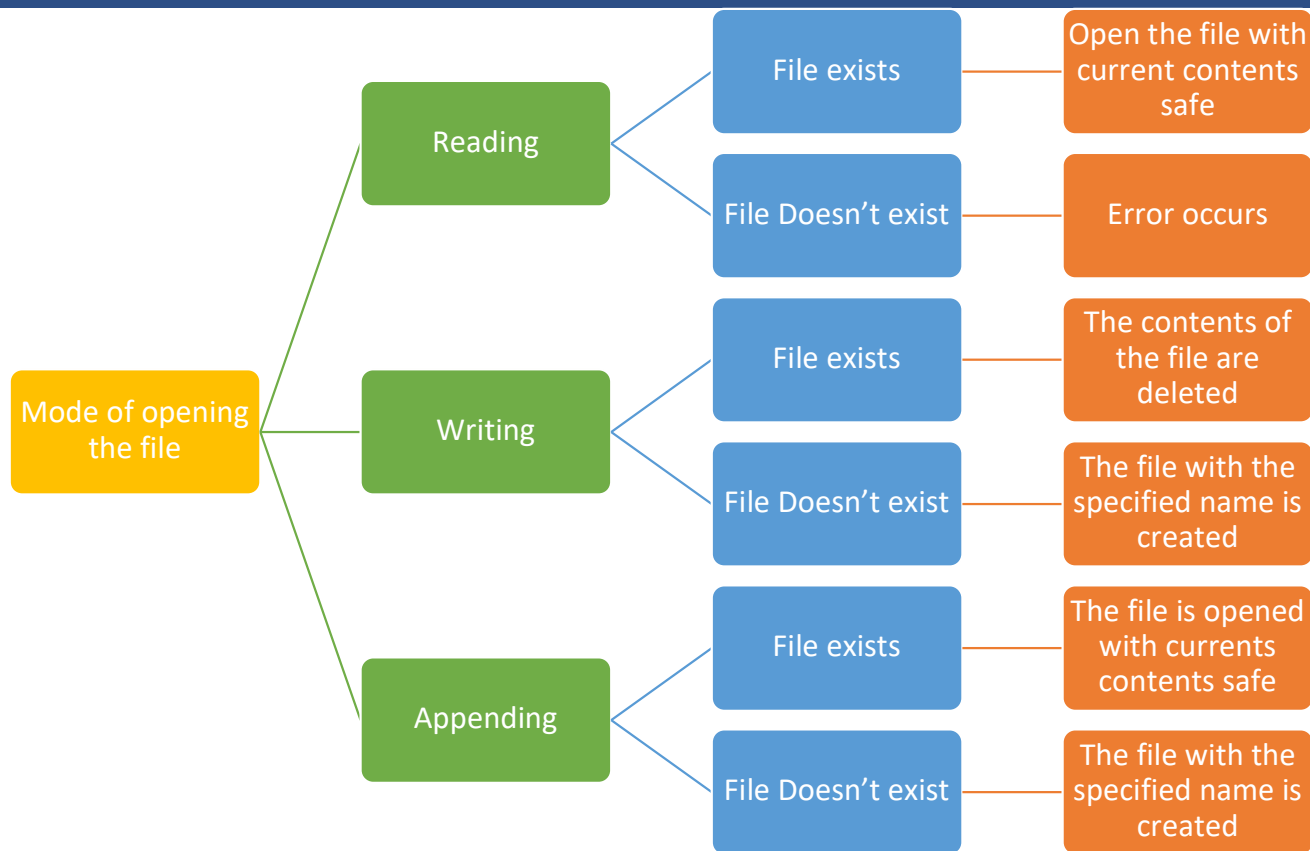
```
FILE *fp;  
fp=fopen("filename", "mode");
```

Declares
variable fp
as a
"pointer to
the data
type FILE"

Opens the file
named "filename"
and assigns an
identifier to the
FILE type pointer fp

r-open the file for reading only
w-open the file for writing only
a-open the file for appending data.

Defining and Opening a File



Closing a File

A file must be closed as soon as all operations on it have been completed.

```
fclose(file_pointer);
```

Closing a file helps in:

- Ensuring all the outstanding information associated with the file is cleared from the buffers
- Preventing accidental misuse of files
- Reopen the same file in other mode

*All files are closed automatically when the program terminates.

Input/Output operations on Files

Getc and putc handles one character at a time.

```
putc(c,fp1);
```

Writes the character contained in variable c to the file associated with file pointer fp1

```
C=getc(fp2);
```

Read a character from the file whose file pointer is fp2

- The file pointer moves by **one character position** for every operation of getc and putc.
- The getc will return an end-of file marker EOF,when end of file has been reached.

Example

```
1  /*Character oriented Read-write operations on File */
2  #include <stdio.h>
3  int main()
4  {
5      FILE *fl;
6      char c;
7      fl= fopen("INPUT", "w"); /* open file for writing */
8
9      while((c=getchar()) != EOF) /*get char from keyboard until CTL-Z*/
10         putc(c,fl); /*write a character to INPUT */
11
12     fclose(fl); /* close INPUT */
13     fl=fopen("INPUT", "r"); /* reopen file */
14
15     while((c=getc(fl))!=EOF) /*read character from file INPUT*/
16         printf("%c", c); /* print character to screen */
17
18     fclose(fl);
19     return 0;
20 } /*end main */
```

"E:\Chapter 12\program1.exe"

```
File management
^Z
File management

Process returned 0 (0x0)   execution time : 13.393 s
Press any key to continue.
```

This PC > New Volume (E:) > Chapter 12			
Name	Date modified	Type	Size
INPUT	20-Jan-21 3:57 PM	File	1 KB
program1.c	20-Jan-21 3:49 PM	C Source	1 KB
program1.exe	20-Jan-21 3:57 PM	Application	29 KB
program1.o	20-Jan-21 3:57 PM	O File	1 KB

This PC > New Volume (E:) > Chapter 12			
Name	Date modified	Type	Size
program1.c	20-Jan-21 3:49 PM	C Source	1 KB

INPUT - Notepad

File Edit Format View Help
File management

Input/Output operations on Files

- Getw and putw-integer oriented functions
- Useful when we deal with only integer data

```
putw(integer,fp);  
getw(fp);
```

Example

```
main()
{
    FILE *f1,*f2,*f3;
    int number,i;
    printf("contents of DATA file\n");
    f1=fopen("DATA","w"); /* Create a DATA file */

    for(i=1;i<=30;i++) /*Read the contents from the user
                        and Write the contents into the DATA file*/
    {
        scanf("%d",&number);
        if(number== -1) break;
        putw(number,f1);
    }

    fclose(f1); /* close the DATA File*/

    f1=fopen("DATA","r");
    f2=fopen("ODD","w");
    f3=fopen("EVEN","w");

    /*Read from DATA File*/
    while((number=getw(f1))!= EOF)
    {
        if(number %2 ==0)
            putw(number,f3); /* Write to EVEN File*/
        else
            putw(number,f2);/* Write to ODD File*/
    }
    fclose(f1);
    fclose(f2);
    fclose(f3);

    f2=fopen("ODD","r"); /* printing the contents of ODD file*/
    printf("\nContents of ODD file\n");
    while((number=getw(f2))!=EOF)
        printf("%4d",number);
    fclose(f2);

    f3=fopen("EVEN","r"); /* printing the contents of EVEN file*/
    printf("\nContents of EVEN file\n");
    while((number=getw(f3))!=EOF)
        printf("%4d",number);
    fclose(f3);
}
```

"E:\Chapter 12\program2.exe"

```
contents of DATA file
11 22 33 44 55 66 77 88 99 -1

Contents of ODD file
 11 33 55 77 99
Contents of EVEN file
 22 44 66 88
Process returned 0 (0x0)   execution time : 8.715 s
Press any key to continue.
```

This PC > New Volume (E:) > Chapter 12					Search Ch...
Name	Date modified	Type	Size		
DATA	21-Jan-21 9:34 AM	File	1 KB		
EVEN	21-Jan-21 9:34 AM	File	1 KB		
INPUT	20-Jan-21 3:57 PM	File	1 KB		
ODD	21-Jan-21 9:34 AM	File	1 KB		
program1.c	20-Jan-21 3:49 PM	C Source	1 KB		
program1.exe	20-Jan-21 3:57 PM	Application	29 KB		
program1.o	20-Jan-21 3:57 PM	O File	1 KB		
program2.c	21-Jan-21 9:34 AM	C Source	2 KB		
program2.exe	21-Jan-21 9:34 AM	Application	30 KB		
program2.o	21-Jan-21 9:34 AM	O File	2 KB		

Input/Output operations on Files

fprintf and fscanf- using mixed data type simultaneously

```
fprintf(fp,"control string",list);
```

Fp is the file pointer associated with a file opened for writing

Control string:- contains output specifications for the items in the list

List: includes variables, constants and strings

```
fscanf(fp,"control string",list);
```

Fp is the file pointer associated with a file opened for reading

Control string:- contains input specifications for the items in the list

List: includes variables, constants and strings

fscanf() returns the number of items that are successfully read. When the end of file is reached it returns EOF

Example

```
#include<stdio.h>
main()
{
    FILE *fp;
    int number,quantity,i;
    float price,value;
    char item[10],filename[10];
    printf("input file name\n"); /* Read the file name for user*/
    scanf("%s",filename);
    fp=fopen(filename,"w");
    printf("enter inventory data \n");
    printf("itemname number price quantity");

    for(i=1;i<=3;i++) /*Read the content from
                        standard input and display to the user */
    {
        fscanf(stdin,"%s %d %f %d",item,&number,&price,&quantity);
        fprintf(fp,"%s %d %.2f %d \n", item,number,price,quantity);
    }
    fclose(fp);
}
```

"E:\Chapter 12\program3.exe"

```
input file name
INVENTORY
enter inventory data
itemname number price quantity
A-1 111 50.5 12
B-1 222 40.3 15
C-1 333 78.35 5

Process returned 0 (0x0)   execution time : 57.269 s
Press any key to continue.
```

inventory - Notepad

File	Edit	Format	View	Help
A-1	111	50.50	12	
B-1	222	40.30	15	
C-1	333	78.35	5	

Example

```
/* Reading from the file and print to the standard output*/
#include<stdio.h>
main()
{
    FILE *fp;
    int number, quantity, i;
    float price, value;
    char item[10], filename[10];
    printf("input file name\n"); /* Read the file name for user*/
    scanf("%s", filename);
    fp=fopen(filename, "r");

    printf("Itemname Number Price Quantity Value\n");
    for(i=1; i<=3; i++)
    {
        fscanf(fp, "%s %d %f %d", item, &number, &price, &quantity);
        value=price*quantity;
        fprintf(stdout, "%s\t%d\t%.2f\t%d\t%.2f\n", item, number, price, quantity, value);
    }
    fclose(fp);
}
```

inventory - Notepad

File Edit Format View Help

```
A-1 111 50.50 12
B-1 222 40.30 15
C-1 333 78.35 5
```

"E:\Chapter 12\program4.exe"

```
input file name
inventory
Itemname Number Price Quantity Value
A-1      111      50.50    12      606.00
B-1      222      40.30    15      604.50
C-1      333      78.35     5      391.75
```

```
Process returned 0 (0x0)   execution time : 4.332 s
Press any key to continue.
```


Error handling during I/O operations

Errors may occur during I/O operations in any of the following scenarios:

1. Trying to read beyond the end-of-file mark.
2. Device overflow
3. Trying to use a file that has not been opened
4. Trying to perform an operation on file ,when the file is opened for another type of operation
5. Opening a file with an invalid filename.
6. Attempting to write a write protected file

2 status inquiry library functions to resolve these errors:

- **Feof()-** used to test for an end of file condition.

```
if(feof(fp))  
printf("End of data");
```

Returns a **nonzero** integer value if **all of the data from the specified file has been read**.

Returns **0 otherwise**

- **Error()-** reports the status of file

```
if(ferror(fp)!=0)  
printf("End of data");
```

Returns a **nonzero** integer if an **error has been detected** during processing.

Returns **0 otherwise**

Example

```
/* Illustration of error handling in file operations*/
```

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

```
int main()
```

```
{  
    FILE *ptr=fopen("PB.dat","w");  
    fprintf(ptr,"COMPUTER");  
    fclose(ptr);
```

```
    ptr=fopen("PB.dat","r");
```

```
    char a;
```

```
    while(1)
```

```
    {  
        a=getc(ptr);  
        iffeof(ptr)  
        {  
            printf("End of file reached");  
            break;
```

```
        }  
        printf("%c\n",a);
```

```
    }
```

```
    fclose(ptr);
```

```
    return 0;
```

```
}
```

PB1.dat - Notepad

File Edit Format View Help

COMPUTER

```
/* ferror() Demonstration*/
```

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

```
#include <stdlib.h>
```

```
int main()
```

```
{  
    FILE *ptr=fopen("PB1.dat","w");  
    char a=getc(ptr);
```

```
    if(ferror(ptr))
```

```
    {  
        printf("Indicates Error");  
    }
```

```
    fclose(ptr);
```

```
    return 0;
```

```
}
```

"E:\Chapter 12\program5.exe"

C

O

M

P

U

T

E

R

End of file reached

Process returned 0 (0x0) execution time : 0.030 s

Press any key to continue.

"E:\Chapter 12\program6.exe"

Indicates Error

Process returned 0 (0x0) execution time : 0.035 s

Press any key to continue.

Random Access to files

ftell()- returns the current position of the file pointer in the given file

N is the relative offset(bytes) of the current position
N bytes have already read or written

`N=ftell(fp);`

Rewind () – resets the position to the start of the file
Helps in reading the file more than once without closing it.

`rewind(fp);`
`N=ftell(fp);`

```
/* Illustration of ftell() and rewind() functions*/
#include <stdio.h>
#include <string.h>

int main()
{
    FILE *ptr=fopen("test1.dat", "w");
    fprintf(ptr, "COMPUTER");
    int position=ftell(ptr);
    printf("Value of 'position' = %d\n", position);

    fprintf(ptr, "sof");
    position=ftell(ptr);
    printf("Value of 'position' before rewind()= %d\n", position);

    rewind(ptr);
    position=ftell(ptr);
    printf("Value of 'position' after rewind()= %d\n", position);

    fclose(ptr);
    return 0;
}
```

test1.dat - Notepad
File Edit Format View Help
COMPUTERsof

"E:\Chapter 12\program7.exe"
Value of 'position' = 8
Value of 'position' before rewind()= 11
Value of 'position' after rewind()= 0
Process returned 0 (0x0) execution time : 0.046 s
Press any key to continue.

Random Access to files

fseek() :used to move the file pointer to a **desired location** with in the file.

Returns **0** upon success, **-1** when an error occurs

Syntax:

```
fseek(file_ptr,offset,position);
```

- Specifies number of positions(bytes) to be moved from the location specified by the **position**.
- Offset may be positive or negative**

Value	Meaning
0	Beginning of file
1	Current position
2	End of file

Examples of fseek():

Statement	Meaning
fseek(fp,0L, 0)	Go to beginning(similar to rewind)
fseek(fp,0L, 1)	Stay at the current position (Rarely used)
fseek(fp,0L, 2)	Go to the end of the file,past the last character of the file
fseek(fp,m,0)	Move to (m+1)th byte in the file
fseek(fp,-m,1)	Go backward by m bytes from the current position
fseek(fp,-m,2)	Go backward by m bytes from the end.

Example

```
//fseek() demonstration
#include <stdio.h>
#include <string.h>

int main()
{
    FILE *ptr=fopen("test2.dat", "w");
    fprintf(ptr, "COMPUTER");
    int position=ftell(ptr);
    printf("before fseek(): Value of current 'position' = %d\n", position);
    printf("After fseek()\n");
    fseek(ptr, 0, 0); /*Go to beginning(similar to rewind)*/
    position=ftell(ptr);
    printf("fseek(ptr, 0, 0) : Value of 'position' = %d\n", position);

    fseek(ptr, 0, 1); /*Stay at the current position */
    position=ftell(ptr);
    printf("fseek(ptr, 0, 1) : Value of 'position' = %d\n", position);

    fseek(ptr, 0, 2); /*Go to the end of the file, past the last character of the file*/
    position=ftell(ptr);
    printf("fseek(ptr, 0, 2) : Value of 'position' = %d\n", position);

    fseek(ptr, 3, 0); /*Move to (m+1)th (4) byte in the file*/
    position=ftell(ptr);
    printf("fseek(ptr, 3, 0) : Value of 'position' = %d\n", position);

    fseek(ptr, -3, 1); /*Go backward by m (3)bytes from the current position*/
    position=ftell(ptr);
    printf("fseek(ptr, -3, 1) : Value of 'position' = %d\n", position);

    fseek(ptr, -3, 2); /*Go backward by m (3)bytes from the end.*/
    position=ftell(ptr);
    printf("fseek(ptr, -3, 2) : Value of 'position' = %d\n", position);

    fclose(ptr);
    return 0;
}
```

"E:\Chapter 12\program8.exe"

before fseek(): Value of current 'position' = 8

After fseek()

fseek(ptr, 0, 0) : Value of 'position' = 0

fseek(ptr, 0, 1) : Value of 'position' = 0

fseek(ptr, 0, 2) : Value of 'position' = 8

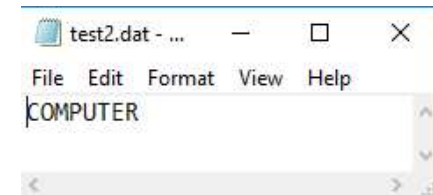
fseek(ptr, 3, 0) : Value of 'position' = 3

fseek(ptr, -3, 1) : Value of 'position' = 0

fseek(ptr, -3, 2) : Value of 'position' = 5

Process returned 0 (0x0) execution time : 0.016 s

Press any key to continue.



Example

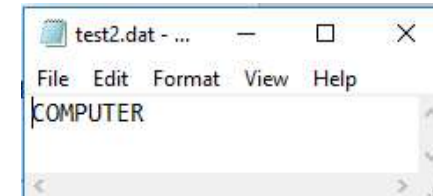
```
/*Write a program to read a text file 'Demo.txt' and
print each word of that file in reverse order.*/

#include <stdio.h>
#include <string.h>

int main()
{
    FILE *ptr=fopen("test2.dat","r");
    char c;
    fseek(ptr,-1,2); // Move the file pointer to the end of the file
    int position=ftell(ptr);
    int size=position+1; // Calculate the size of the file

    while(size)
    {
        c=getc(ptr); // Read the character and print on console
        printf("%c",c);
        fseek(ptr,-2,1); // Shift the pointer to the previous location
        size--;
    }

    fclose(ptr);
    return 0;
}
```



"E:\Chapter 12\Program10.exe"

RETUPMOC

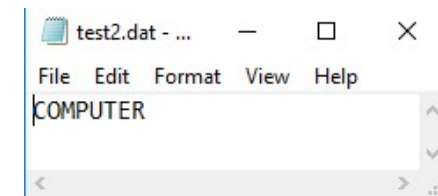
Process returned 0 (0x0) execution time : 0.023 s
Press any key to continue.

Example

```
/*Write a program to print last n characters of a file.
Use the function fopen(), fclose(), ftell(),
fseek() and rewind().*/

#include <stdio.h>
#include <string.h>

int main()
{
    FILE *ptr=fopen("test2.dat","r");
    char c;
    int n;
    printf("enter the number of characters to print ");
    scanf("%d",&n);
    fseek(ptr,-n,2);
    printf("File pointer is at %d position \n",ftell(ptr));
    while(n)
    {
        c=getc(ptr); // Read the character and print on console
        printf("%c",c);
        fseek(ptr,0,1);
        n--;
    }
    fclose(ptr);
    return 0;
}
```



"E:\Chapter 12\Program16.exe"

```
enter the number of characters to print 4
File pointer is at 4 position
UTER
Process returned 0 (0x0)   execution time : 2.660 s
Press any key to continue.
```


Command Line arguments

- C programming language gives the programmer the provision to add parameters or arguments inside the main function.
- Command line arguments are simply arguments that are specified after the name of the program in the system's command line, and these argument values are passed on to your program during program execution.

Components of Command Line Arguments

- There are **2** components of Command Line Argument in C:
- **argc**: It refers to “argument count”. It is the first parameter that we use to store the number of command line arguments. It is important to note that the value of argc should be greater than or equal to 0.
- **argv**: It refers to “argument vector”. It is basically an array of character pointer which we use to list all the command line arguments.

The basic syntax is:

```
int main( int argc, char *argv[] )  
{  
    // BODY OF THE MAIN FUNCTION  
}
```

or it can also be written as

```
int main( int argc, char **argv[] )  
{  
    // BODY OF THE MAIN FUNCTION  
}
```

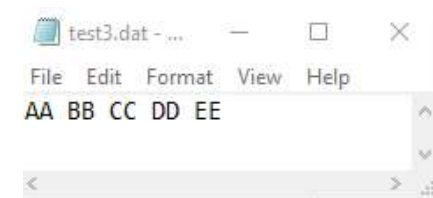
Example

```
program9.c X
1  /* Demonstration of command line arguments*/
2  #include<stdio.h>
3  int main(int argc, char *argv[])
4  {
5
6      FILE *fp;
7      int i;
8      char word[15];
9      fp=fopen(argv[1], "w"); /*open file with name argv[1];*/
10     printf("\n No of arguments in command line =%d :\n", argc);
11     for(i=2; i<argc; i++)
12         fprintf(fp, "%s ", argv[i]);
13     fclose(fp);
14
15     /* writing contents on to the screen*/
16     printf("Contents of %s file are : " , argv[1]);
17     fp=fopen(argv[1], "r");
18     for(i=2; i<argc; i++)
19     {
20         fscanf(fp, "%s", word);
21         printf("%s ", word);
22     }
23     fclose(fp);
24     return 0;
25 }
26
```

Administrator: Command Prompt

E:\Chapter 12>program9 test3.dat AA BB CC DD EE

No of arguments in command line =7 :
Contents of test3.dat file are :AA BB CC DD EE
E:\Chapter 12>



Example

```
#include<conio.h>
int main() {

FILE *fp;
char another='y';

struct blood {
    char name[50];
    char adr[50];
    int age;
    int bld;
} b;

fp=fopen("BLOODBANK.DAT","w");

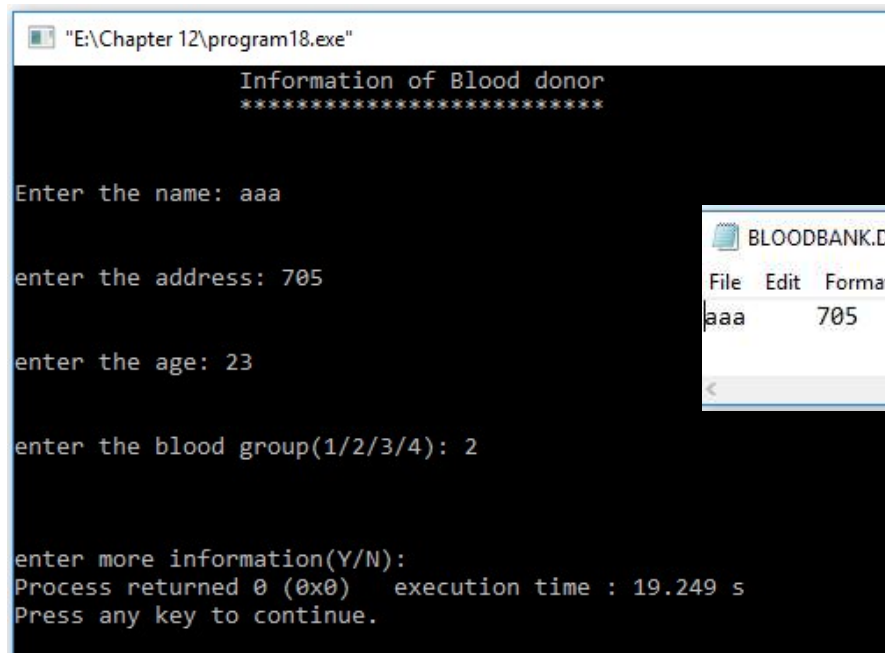
if(fp==NULL) {
printf("cannot open target file!\n");
//exit();
}

while(another=='Y' || another=='y') {
printf("\t\t\tInformation of Blood donor\n");
printf("\t\t\t*****\n\n\n");
printf("Enter the name: ");
scanf("%s",b.name);
printf("\n\nenter the address: ");
scanf("%s",b.adr);
printf("\n\nenter the age: ");
scanf("%d",&b.age);
printf("\n\nenter the blood group(1/2/3/4): ");
scanf("%d",&b.bld);
```

```
fprintf(fp,"%s\t%s\t%d\t%d",b.name,b.adr,b.age,b.bld);

printf("\n\n\nenter more information(Y/N): ");
fflush(stdin);

another=getch();
}
fclose(fp);
}
```



```
"E:\Chapter 12\program18.exe"

Information of Blood donor
*****

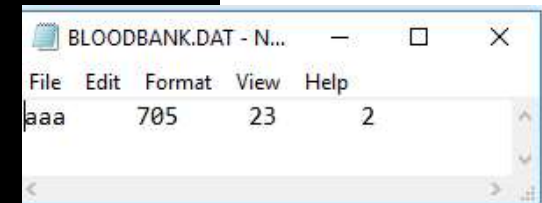
Enter the name: aaa

enter the address: 705

enter the age: 23

enter the blood group(1/2/3/4): 2

enter more information(Y/N):
Process returned 0 (0x0)   execution time : 19.249 s
Press any key to continue.
```



File	Edit	Format	View	Help
aaa	705	23	2	

Example

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

FILE *fp;
char ch;

struct blood {
    char name[50];
    char adr[50];
    int age;
    int bld;
}b;

fp=fopen("BLOODBANK.DAT","r");

if(fp==NULL) {
printf("cannot open source file!\n\n");
//exit();
}

while(fscanf(fp,"%s\t%s\t%d\t%d",&b.name,&b.adr,&b.age,&b.bld)!=EOF)
if(b.age<25 && b.bld==2) {
printf("\n%s\t %s\t%2d\t %d",b.name,b.adr,b.age,b.bld);
}
fclose(fp);

return 0;
}
```

"E:\Chapter 12\Program17.exe"

```
aaa      705    23      2
Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

Thank You.