- **Bit Fields**

While we're on the subject of structures, we might as well look at bitfields. They can only be declared inside a structure or a union, and allow you to specify some very small objects of a given number of bits in length. Their usefulness is limited and they aren't seen in many programs, but we'll deal with them anyway.

If in a program a variable is to take only two values 1 and 0, we really need only a single bit to store it. Similarly, if a variable is to take value from 0 to 3, then two bits are sufficient to store these values and if a variable is to take values from 0 through 7, then three bits will be enough, and so on.

Example:

```
struct student
{
unsigned pupil:1;
unsigned trade:2;
};
struct student s;
```

** **Bitfield Program**

```
#include <stdio.h>
#include <string.h>
/* define simple structure */
struct
{
unsigned int widthValidated;
unsigned int heightValidated;
} status1;

/* define a structure with bit fields */
struct {
unsigned int widthValidated : 1;
unsigned int heightValidated : 1;
} status2;

int main( )
 {
printf( "Memory size occupied by status1 : %d\n", sizeof(status1));
```

```
printf( "Memory size occupied by status2 : %d\n", sizeof(status2));
return 0;
}
```

OUTPUT

Memory size occupied by status1: 8
Memory size occupied by status1: 4

File Management

- **File**

We frequently use files for storing information which can be processed by our programs. In order to store information permanently and retrieve it we need to use files. Files are not only used for data. Our programs are also stored in files.

File pointer is a pointer to a structure of type **FILE.** It points to information that defines various things about the file, including its name, status, and the current position of the file.

Commonly Used C file-system Function

Name	Function
fopen()	Opens a file
fclose()	Closest a file
putc()	Writes a character to a file
fputc()	Same as **putc()**
getc()	Reads a character from a file
fgetc()	Same as **getc()**
fgets()	Reads a string from a file
fputs()	Writes a string to a file
fseek()	Sets the position to a desired point in the file
ftell()	Returns the current file position
fprintf()	Writes a set of data values to a file
fscanf()	Reads a set of data values from a file
feof()	Returns true if end-of-file is reached
ferror()	Returns true if an error has occurred
rewind()	Sets the position to the beginning of the file
remove()	Erases a file
fflush()	Fluhes a file

Opening Files:

Before performing the any I/O file, the file must be opened. While opening the file, the following are specified:

- The name of the file
- A file can be opened in various modes. This mode decides the read/write operation with the data/result file. Table shows the legal values for mode

Mode	Meaning
r	Opens a text file for reading
w	Creating a text file for writing
a	Appends to a text file
rb	Opens a binary file for reading
wb	Opens a binary file for writing
ab	Appends to a binary file
r+	Opens a text file for read/write
w+	Create a text file for read/write
a+	Appends or create text file for read/write
rb+	Opens a binary file for read/write
wb+	Create a binary file for read/write
ab+	Appends or Create a binary file for read/write

To open a file, the function *fopen ()* is used. It is as follow

FILE *fp=fopen ("file name","mode");

fp is a file pointer.

File –Handling Functions:

fopen()

Declaration : **FILE *fp;**

fp=fopen(file name,mode);

Where fp is a file pointer. ***fopen()*** return a pointer to the opened file stream. The parameter **filename** is the name of the file to be opned.

fprintf()

Declaration by example: **int i;**

fprintf(fp,"%d",i);

This is the same as **printf** except that a pointer to a file strteam must be specified. The function **fprintf()** writes to the file associated with the file stream .

fscanf()

Declaration by example: **int i;**

fscanf(fp,"%d",&i);

The function **fscanf()** reads from the file associated with the file stream. The arguments passed are similar to **scanf()** ,except that the file stream must be specified.

fclose()

Declaration: **fclose(fp);**

The buffer associated with the stream is flushed before being closed. System-allocated buffers are also released on closing. **Stdin, stdout** are also closed using **fclose.** Only **freopen** can reopen these streams.

fgetc()

Declaration: **fgetc(fp);**

The function **fgetc()** returns the next character in the named input stream. It is a function style of the macro **getc.**

fputc()

Declaration: **fputc(int c,fp);**

The function **fputc()** outputs the value of the character **c** to the specified stream.