Chapter - 11 : File Processing

Formatted File Input and Output

- Formatted file input/output deals with text and numeric data that is formatted in a specific way.
- It is directly analogous to formatted keyboard input and screen output done with the printf() and scanf() functions.

Formatted File Output

- Formatted file output is done with the library function fprintf().
- The prototype of fprintf() is in the header file STDIO.H, and it reads as follows:

  int fprintf(FILE *fp, char *fmt, ...);

- The first argument is a pointer to type FILE.
- To write data to a particular disk file, pass the pointer that was returned when the file was opened with fopen().
- The second argument is the format string.
- The format string used by fprintf() follows exactly the same rules as printf().
- In other words, in addition to the file pointer and the format string arguments, fprintf() takes zero, one, or more additional arguments.
- This is just like printf( ). These arguments are the names of the variables to be output to the specified stream.
- Remember, fprintf( ) works just like printf( ), except that it sends its output to the stream specified in the argument list.

Formatted File Input

- For formatted file input, use the fscanf() library function, which is used like scanf() , except that input comes from a specified stream instead of from stdin.
- The prototype for fscanf() is:

  int fscanf(FILE *fp, const char *fmt, ...);

- The argument fp is the pointer to type FILE returned by fopen(), and fmt is a pointer to the format string that specifies how fscanf() is to read the input.
- The components of the format string are the same as for scanf().
- Finally, the ellipses (...) indicate one or more additional arguments, the addresses of the variables where fscanf() is to assign the input.

Example :- Program to write the rollno , name , marks of 3 subject of 3 students in the file.

```
#include<stdio.h>
void main( )
{
 FILE *fp;
 int rollno,
 char name[30];
 int m1,m2,m3;
 fp=fopen("Stud.txt","w");
 printf("Enter the RollNo Name M1 M2 M3");
 scanf("%d %s %d %d %d", &rollno, name , &m1, &m2, &m3);
 fprint(fp, "%d %s %d %d %d\n", rollno, name , m1, m2, m3);
 printf("Enter the RollNo Name M1 M2 M3");
 scanf("%d %s %d %d %d", &rollno, name , &m1, &m2, &m3);
 fprint(fp, "%d %s %d %d %d\n", rollno, name , m1, m2, m3);
 printf("Enter the RollNo Name M1 M2 M3");
 scanf("%d %s %d %d %d", &rollno, name , &m1, &m2, &m3);
 fprint(fp, "%d %s %d %d %d\n", rollno, name , m1, m2, m3);
 fclos(fp);
}
```

Program to read data from file print to the screen.

```
#include<stdio.h>
void main()
{
 FILE *fp;
 int rollno,
 char name[30];
 int m1,m2,m3;
 fp=fopen("Stud.txt","r");
 while(1)
 {
     fscanf(fp, "%d %s %d %d %d", &rollno, name , &m1, &m2, &m3);
     if (feof(fp)) break;
    printf( "%d %s %d %d %d\n", rollno, name , m1, m2, m3);
 }
 fclos(fp);
}
```

Direct File Input and Output

- The direct file I/O most often when one has to save data to be read later by the same or a different C program.
- Direct I/O is used only with binary mode files.
- With direct output, blocks of data are written from memory to disk.
- Direct input reverses the process: A block of data is read from a disk file into memory.
- For example, a single direct output function call can write an entire array of type double to disk, and a single direct input function call can read the entire array from disk back into memory.
- The direct I/O functions are fread( ) and fwrite( ).

The fwrite( ) Function

- The fwrite() library function writes a block of data from memory to a binary mode file.
- Its prototype in STDIO.H is: **int fwrite(void *buf, int size, int count, FILE *fp);**
- The argument **buf** is a pointer to the region of memory holding the data to be written to the file.
- The pointer type is void; it can be a pointer to anything.
- The argument size specifies the size, in bytes, of the individual data items, and count specifies the number of items to be written.
- The argument **fp** is, of course, the pointer to type FILE, returned by fopen( ) when the file was opened.
- The fwrite() function returns the number of items written on success; if the value returned is less than count, it means that an error has occurred.

Example

```
#include<stdio.h>
void main()
{
 FILE *fp;
 char name[30]="Ajay Kumar";
 fp=fopen("Stud.txt","wb");
 fwrite(str, sizeof(str), 1, fp);
 fclos(fp);
}
```

The fread() Function

- The fread() library function reads a block of data from a binary mode file into memory.
- Its prototype in STDIO.H is: **int fread(void *buf, int size, int count, FILE *fp);**
- The argument buf is a pointer to the region of memory that receives the data read from the file.
- As with fwrite(), the pointer type is void.
- The argument size specifies the size, in bytes, of the individual data items being read, and count specifies the number of items to read.
- Note how these arguments parallel the arguments used by fwrite( ).
- Again, the sizeof() operator is typically used to provide the size argument.

- The argument fp is the pointer to type FILE that was returned by fopen( ) when the file was opened.
- The fread( ) function returns the number of items read; This can be less than count if end-of-file was reached or an error occurred.

Example

```
#include<stdio.h>

void main()
{
  FILE *fp;

  char name[30];

  fp=fopen("Stud.txt","rb");

  fread(str,  30 , 1,  fp);

  printf(("%s", str);

  fclos(fp);
}
```