```
{
        FILE *fp;
        int c;
        fp = fopen("in.dat", "r");
        if (fp == NULL)
        {
                printf("Error in opening file\n");
                exit(1);
        }
        while ((c = getc(fp)) != EOF)
                printf("Char. read = %c\n", c);
            fclose(fp);
}
```

Sample Run

```
Char. read = B
Char. read = I
Char. read = T
Char. read =
```

The program shown above assumes that the input file *in.dat* already exists and contains the text *'BIT*. As explained already, *getc* reads characters one by one from *in.dat* and writes it on the console. The last character read is a file terminator not printable.

A much more useful and meaningful example is shown in Program 1.41. This program counts the number of characters, words, and lines in a given text.

Program 1.40
Count of characters, words, and lines.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
     FILE *fp;
     char fname[25];
     int c;
     int linecount = 0;
     int wordcount = 0;
     int charcount = 0;

     printf("Enter the file name : ");
     scanf("%s", fname);
     fp = fopen(fname, "r");
     if (!fp)
     {
```

```
        printf("Error in opening file <press Enter key>\n");
        fflush(stdin);
            getchar();
            exit(1);
    }
    while ((c = getc(fp)) != EOF)
    {
            charcount++;

            switch(c)
            {
                    case '\n': linecount++;
                               wordcount++;
                               break;

                    case  ' ': wordcount++;
                               break;
            }
    }
    printf("No. of Chars. : %d\n", --charcount);
    printf("No. of lines  : %d\n", linecount);
    printf("No. of words  : %d\n", wordcount);
    getchar();
    fclose(fp);


}
```

Sample Run

Contents of in.dat

```
Bangalore Institute of Technology is
in the heart of Bangalore city.
D:\DS-Book>file1
Enter the file name : in.dat
No. of Chars. : 69
No. of lines  : 2
No. of words  : 11
```

Following points are worth mentioning:

(1) File names can be given at run time also (*fname* does this job).

(2) The file *fname* is opened in read mode.

(3) The variable `charcount` is incremented whenever a character is read. Word count is done based upon a blank character and line count is based on a new line character '\n'.

(4) The input file is not altered.

1.13.4 Stream Functions – "w" mode

When a file is opened in "w" mode, data can be written into it from the beginning of the file. This mode creates a new file if doesn't exist already. If it exists its old contents is deleted. Therefore, you must be careful when you specify the output file name.

Program 1.41 demonstrates write mode by first reading some text from the keyboard using *getchar()* until the user presses *<ctrl-Z>*. All characters entered are written to the file named as *exp.dat*. The same file is opened in "*r*" and its contents displayed in the usual manner. Notice that we have two file pointers: `infp` for reading and `outfp` for writing (you can manage with single file pointer also. See the Program 1.42).

Program 1.41
Read from keyboard and write into a file

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
     FILE *infp, *outfp; /* i/o file pointers */
     int c;
     outfp = fopen("exp.dat","w");

     /* read from keyboard and write into output file */
     while ((c = getchar()) != EOF)
          putc(c,outfp);

     infp = fopen("exp.dat","r");
     if (infp == NULL)
     {
          printf("Error in opening file\n");
          exit(1);
     }

     /* read from input file & display on the console */
     while (( c = getc(infp)) != EOF)
          putc(c, stdout);

     fcloseall();
}
```

1.13.5 Stream Functions – "a" mode

The **"a" mode** or **append mode** is almost same as write mode with a difference that you can append data to an existing at its end. As given in Table 1.11, the file pointer is positioned at the end of the file in append mode so that data can be added to an existing file. If the specified file in *fopen()* is not in the disk, a new file will be created.

Let us show the working of "*a*" mode with an example program and is shown in Program 1.42. This program assumes that the data file exists already and the text to be appended will be read from the user. The contents of the file before and after appending are displayed in the normal manner.

Program 1.42
Appending text to a file

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
     FILE *fp;
     int c;

     printf("Text before appending:\n");
     fp = fopen("exp.dat", "r");
     if (fp == NULL)
     {
          printf("Error in opening file\n");
          exit(1);
     }
     while ((c = fgetc(fp)) != EOF)
          fputc(c, stdout);

     fp = fopen("exp.dat", "a");
     printf("Enter text for appending:\n");
     while ((c = getchar()) != EOF)
          fputc(c,fp);
     fclose(fp);

     printf("Text after appending:\n");
     fp = fopen("exp.dat", "r");
     if (fp == NULL)
     {
          printf("Error in opening file\n");
          exit(1);
     }
```