CHAPTER 10

Data File:

prepared by: Shiva pd. Mahato

Introduction

Data File:

- A file is a set of information or group of related data stored in a particular place on the memory device of the computer. In short, a file is a collection of records.
- A real life applications involve large amounts of data and in such situations the console oriented I/O operations posses two major problems:
- First, it becomes cumbersome and time consuming to handle huge amount of data through terminals.
- Second, when doing I/O using terminal, the entire data is lost when the program is terminated or computer is turned off. Therefore, it becomes necessary to store data on a permanent storage (the disks) and read whenever necessary, without destroying the data.
- Therefore ,most of the application program which processes large volume of data that should be permanently stored in data files. In order to store,information permanently for future use and retrieve, we need to use data files.
- File helps us to access the data i.e. read the data from the memory location and writing the data, rearranging it etc whenever necessary.

Stream in C:

- In C, the standard streams are termed as pre-connected input and output channels between a text terminal and the program (when it begins execution). Therefore, stream is a logical interface to the devices that are connected to the computer.

Data file

- Stream is widely used as a logical interface to a file where a file can refer to a disk file, the computer screen, keyboard, etc. Although files may differ in the form and capabilities, all streams are the same.
- The three standard streams in C languages are- standard input (stdin), standard output (stdout) and standard error (stderr).
- Standard input (stdin): Standard input is the stream from which the program receives its data. The program requests transfer of data using the read operation.
- However, not all programs require input. Generally, unless redirected, input for a program is expected from the keyboard.
- Standard output (stdout): Standard output is the stream where a program writes its output data. The program requests data transfer using the write operation. However, not all programs generate output.

- Standard error (stderr): Standard error is basically an output stream used by programs to report error messages or diagnostics. It is a stream independent of standard output and can be redirected separately. No doubt, the standard output and standard error can also be directed to the same destination.

Data file

- **File Pointer** :
- When working with a data file, the first step is to established a buffer a, where information is temporarily stored while being transferred between computer's memory and the data file.
- This buffer area allows information to be read from or written to the data file more rapidly than would otherwise be possible. The buffer area is established by
- FILE *ptr;
- Where FILE (upper letters required) is a special structure type(new data type) that established the buffer area and ptr is a pointer variable that is indicates the beginning of the buffer area. This pointer ptr is often referred to as a stream pointer .
- The FILE object contains all the information about stream like current position, pointer to a buffer, error and EOF (end of file).
- Before writing information to a file or read it, we must open the file first.
- C supports a number of functions that have the ability to perform basic file operations which include
- opening the file
- reading data from a file
- writing data to a file
- closing a file

Data file

Opening a file:

- A data file must be opened before it can be processed. Since file can be used in different ways i.e. for reading or writing to or both. In C programming, the mode of file is specified while opening the data file and library function *fopen()* defined in *stdio.h* is used to open a file.
- The general format for opening a file is given below.
- FILE *ptr
- ptr=fopen("filename", "mode");
- Where fopen() library function is used to open a specified file on the disk and ptr is the pointer variable. Mode is the type of operations to be performed or purpose of opening the file. The mode of the file may be one of the following.
- "r"- opens an existing file in read only mode, if the file already exists otherwise returns NULL
- "w"- If the file already exists, then existing file will be opened in write mode and data in the is destroyed. If the file doesn't exits then new file will be created in its place.
- "a"- opens the existing file for appending (i.e. for adding new information at the end of the file). A new file will be created if the file with the specified filename does not exist.

Data file

- "**r+**"- opens existing file for both reading and writing. If file does not exist, a new file will be created.
- "**w+**"- opens a new file for both reading and writing. If the file already exists, it will be destroyed and new file will be created in its place.
- "**a+**"- opens the existing file for both reading and appending. A new file will be created if the file with the specified filename doesn't exist.
- During a write to a file, the data and the information is not put on the storage device immediately, it is stored in a buffer. Thus, whenever the buffer is full, all its contents are actually written to the storage device i.e. memory or disk. This process of emptying the buffer by writing its contents to disk is called flushing the buffer.
- Thus each and every file must be closed after opened once and the closing of the file flushes the buffer and releases the space taken by the FILE structure which is returned by fopen().

fclose():

- The fclose() function is used to close the file for reuse and the file pointer will be unlinked from the system name. Note that the argument form fclose() is a file pointer not a file name. The closing of the file helps to break the link between buffer and file and also prevents any accidental misuse of the file.
- Syntax for fclose() is
- fclose(ptr);
- The fclose() fuction returns 0 if the close operation is successful, otherwise returns -1, it is good practice to check for successful close operation because portion of the file can be lost if it is not close properly.

Data file

File handling I/O Functions:

- The following are file handling I/O functions used to perform the input output operation in file:
- **Character I/O:** getc(),putc(),fgetc(),fputc()
- **String I/O:** fgets(), fputs()
- **Word I/O:** getw(),putw()
- **Formatted I/O:** fprintf(),fscanf()
- fgetc()- used to read single character from the specified file and returns the value of the end of the file(EOF) or an error is encountered.The getc() is a micro while fgetc() is the true function in the stdio.h library.
- Syntax:
- char ch;
- ch=fgetc(fp);
- Where ch is a character variable and fp is the FILE pointer.
- fputc()-used to write a single character in to the file. The format of the fputc() is same as that the putc(). The only difference is that the putc is a micro and fputc() is a true function in stdio.h library.
- Syntax: fputc(ch,fp);
- Where ch is a character variable and fp is the FILE pointer.

Data file

- fputc()-used to write a single character in to the file. The format of the fputc() is same as that the putc(). The only difference is that the putc is a micro and fputc() is a true function in stdio.h library.
- Syntax: fputc(ch,fp);
- Where ch is a character variable and fp is the FILE pointer.
- fgets()- used to read a line of text with specified number of characters from the file.
- Syntax: fgets(strptr,n,fp);
- Where strptr is a character array, n is the number of characters and fp is a FILE pointer.
- fputs()- used to write a line of text to a file
- Syntax;
- fputs(strptr,fp);
- Where strptr is a character array and fp is a FILE pointer.
- feof()-detects the end of the file. If end of file is detected then it returns value 1 otherwise 0.
-

Data file

- **Formatted file handling functions**
- fprintf()-> used to write the formatted data to a file
- Syntax: fprintf(fp,"control string",list of variable separated by commas);
-
- fscanf()->used to read formatted data from the file
- Syntax: fscanf(fp,"control string", list of variable with address sign);
- //WAP to enter a line of text and store them in a file, then read and displays the contents of that file.
- #include<stdio.h>
  #include<process.h>
- void main()
- {
- char ch;
- FILE *fp;
- Fp=fopen("test.txt","w");
- If(fp==NULL)
- {
- printf("the file can't be created");
- exit(1);
- }

Data file

```
while((ch=getchar())!='\n')
{
fputc(ch,fp);
}
fclose(fp);
fp=fopen("test.txt","r");
while(!eof(fp))
{
ch=fgetc(fp);
putchar(ch);
}
fclose(fp);
getch();
}
//WAP to read a line of text from the data file and display it on the screen
#include<stdio.h>
void main()
{
FILE *fp;
fp=fopen("sample.txt","r");
if(fp==NULL)
{
printf("can't open the file");
exit(1);
}
```

Data file

```
                do
{
      putchar(c=fgetc(fp));
}     while(c!=EOF);

fclose(fp);
getch();
}
 //WAP to count characters, spaces, tabs and newline in a file
 #include<stdio.h>
#Include<process.h>
void main()
{
char ch;
int noc=0,nol=0,not=0,nos=0;
FILE *fp;
fp=fopen("test.txt","r");
ff(fp==NULL)
{
printf("file can't be opened");
exit(0);
}
while(!feof(fp))
{
ch=fgetc(fp);
```

Data file

```
if(ch==' ') nos++;
if(ch=='\n') nol++;
if(ch=='\t')  not++;
noc++;
}
fclose(fp);
printf("no of chars=%d tabs=%d space=%d lines=%d",noc,not,nos,nol);
getch();
}
//WAP to read the names of n persons and store them in the file, read that file and display.
#include<conio.h>
#include<stdio.h>
#include<process.h>
void main()
{
 FILE *fp;
 int i,n;
 char a[50][20],b[20];
 clrscr();
 fp=fopen("da.txt","w");
 if(fp==NULL)
 {printf("File cannot be opened\n");
   Exit(1);
 }
```

Data file

```
printf("enter the number of data\n");
 scanf("%d",&n);
  for(i=0;i<=n-1;i++)
  {
  fflush(stdin);
  gets(a[i]);
  fputs(a[i],fp);
  }
fclose(fp);
fp=fopen("da.txt","r") ;
printf(" the contents read fro file are\n");
while(!feof(fp))
{fgets(b,20,fp);
 puts(b);
 printf("\n");
}
getch();
}
```

Data file

```
//WAP to copy the content of one file into another file
#include<stdio.h>
#include<process.h>
void main()
{
char ch;
FILE *fs,*fd;
fs=fopen("test.txt","r");
fd=fopen("test1.txt","w");
if(fs==NULL || fd==NULL)
{
printf("some error occurred");
exit(1);
}
while(!feof(fs))
{
ch=fgetc(fs):
fputc(ch,fd);
}
fclose(fs);
printf("file's content  is copied from source file to destination filoe\n");
while(!feof(fd))
{
ch=fgetc(fd);
putchar(ch);
}
fclose(fd);
getch();
}
```

Data file

```
#include <stdio.h>
#include <conio.h>
void main()
{
FILE *fp1, *fp2, *fopen();
int ca, cb;
char fname1[40], fname2[40] ;
printf("Enter first filename:") ;
gets(fname1);
printf("Enter second filename:");
gets(fname2);
fp1 = fopen( fname1,  "r" );       /* open for reading */
fp2 = fopen( fname2,  "r" ) ;      /* open for writing */
if ( fp1 == NULL )      /* check does file exist etc */
{
printf("Cannot open %s for reading \n", fname1 );
exit(1);    /* terminate program */
}
```

Data file

```
else if ( fp2 == NULL )
{
printf("Cannot open %s for reading \n", fname2 );
exit(1);    /* terminate program */
}
else   /* both files opened successfully  */
{
ca  =  getc( fp1 ) ;
cb  =  getc( fp2 ) ;
while ( ca != EOF   &&   cb != EOF   && ca == cb  )
{
ca  =  getc( fp1 ) ;
cb  =  getc( fp2 ) ;
}
if (  ca == cb )
printf("Files are identical \n");
else if ( ca !=  cb )
printf("Files differ \n" );
fclose ( fp1 );
fclose ( fp2 );
}
getch();
}
```

Data file

- Write a program to input a line of text and store them in file. Read the content of the file and determine the number of vowels, number of consonants, number of whitespace number of digits and other characters.

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
void main()
{
 int wh=0,cons=0,oth=0,vow=0,num=0;
 char ch;
 FILE *fp;
clrscr();
 fp=fopen("data.txt","w");
 printf”enter a line of text\n”);
 while((ch=getchar())!='\n')
 {
  fputc(ch,fp);
 }
 fclose(fp);
```

Data file

```
fp=fopen("data.txt", "r");
ch=fgetc(fp);
while(!feof(fp))
{  if((ch>=65 &&ch<=90)||(ch>=97 &&ch<=122))
   {  if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u')
      { vow++;
      }
     else
      { cons++;
      }
   }
   else if(ch==' ')
   { wh++;
   }
   else if(ch>=48 &&ch<=58)
   { num++;
   }
   else
   { oth++;
   }
  ch=fgetc(fp);
}
```

Data file

- fclose(fp);
- printf(" no ofvowels=%d\t no of spaces=%d\t no of consonant=%d\t number= %d\t other =%d\t ",vow,wh,cons,num,oth);
- getch();
- }
- Output:
- enter a line of text
- this is @#program 12
- no of vowels= 4 no of spaces=2 no of consonant=9 number= 2 other =2
- **fwrite() and fread() function:**
- The fwrite() function is used to write a binary data of a given file, it is due used to write one or more structures to a given file.
- The general format of the fwrite() function is :
- fwrite(ptr,size, n,fptr);
- Where ptr is a pointer is the first structure to be written size of the size (in byte) of each structures, n is the number of structures to be written and fptr is the file pointer to the file. The ptr may be point to a variable of any types (from a single character to a structure).
- The size would give the members of bytes in each item to be written. The fwrite() function always returns the numbers of item actually written to the file whether a not of end of the file or an error is encountered.

structure

- //WAP to read name, address, contact number of 5 persons and write it into a file and read and display the contents.
- #include<stdio.h>
- #include<process.h>
- struct
- {
- char name[20];
- char address[20];
- int contact_no;
- };
- void main()
- {
- int i;
- struct person[5], p;
- FILE *fp;
- fp=fopen("student.txt","w");
- if(fp==null)
- printf("file can't open");
- printf("\nenter name,address and contact number of five persons");
- for(i=0;i<5;i++)
- {
- scanf("%s%s%d",person[i].name,person[i].address,&person[i].contact_no);
- fwrite(&person[i], sizeof(person[i],1,fp);
- }

structure

- fclose(fp);
- fp=fopen(“student.txt”,”r”);
- while(fread(&p,sizeof(p),1,fp)==1) //if records is not found it return 0
- {
- printf(“%s\t%s\t%d\n”p.name,p.address,p.contact_no);
- }
- fclose(fp);
- getch();
- }
- //WAP to write and read the information about the players name, age and run. Use fread() and fwrite() functions . User should press y to continue enter the information of next players and N for termination.
-
- #include<stdio.h>
- #include<process.h>
- struct player
- {
- char name[20];
- char age[20];
- int runs;
- };

structure

```
void main()
{
int i;
char next='y';
struct player p;
FILE *fp;
fp=fopen("player.dat","w");
if(fp==null)

{printf("file can't open");
  extit(1);
}
printf("\nenter palyers name, age and runs made");
while(next=='y')
{
scanf("%s%d%d",&p.name,&p.age,&p.runs);
fwrite(&p, sizeof(p),1,fp);
printf("\n do you want to add next players information\n");
fflush(stdin);
next=getche();
}
```

Data file

- fclose(fp);
- fp=fopen(“student.dat”,”r”);
- while(fread(&p,sizeof(p),1,fp)==1)
- {
- printf(“%s\t%d\t%d\n”,p.name,p.age,p.runs);
- }
- fclose(fp);
- getch();
- }
- //WAP to read name, roll and marks of 5 students and sort and store them in file in ascending order. Read the contents of the file.
- #include<conio.h>
- #include<stdio.h>
- struct stdrec
- {
- char name[10],address[20],
- int roll;
- };

Data file

```
struct stdrec std[5],tem;
void main( )
{ FILE *fp;
  fp=fopen(" student.txt", "w");
 for (i=0;i<4;i++)
 { printf("Enter name address and roll of student");
  scanf("%[^\n]%[^\n]%d",std[i]name,std[i].address,&std[i].roll);
   flush(stdin);
  }
   for (i=0;i<4;i++)
  {   for (j=i+1;j<4;j++)
       {   if(strcmp(std[i].name,std[i].name)>0)
           { tem=std[i];
               std[i]=std[j];
               std[j]=tem;
            }
       }
  }
 for (i=0;i<4;i++)
 {
   fwrite(&std[i],sizeof(std[i]),1,fp);
  }
```

Data file

- fclose(fp);
- fp=fopen(“ student.txt”, “r”);
- printf(“The name address and roll of student in alphabetical order are as\n”);
- while(fread(&std,sizeof(std),1,fp)==1)
- {
- printf(“%s%s%d”, sd.name,sd.address,sd.roll);
- }
- getch( );
- }
- **Why to use file?**
- The first and foremost reason to use the file is the need of permanence storage of data that means to preserve t he o/p of any program for future use, which is not possible without files.
- Since the message or values printed with the statements like printf(), putchar() are never available for future use, the only solution in that case is to write the o/p in files.
- Similarly if a lot of data to be inputted in a program again and again for the repeated execution of that program in that case, all input data can be once written in a file and then that file can be easily used as a input file.
- C supports a large number 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.

Error handling in file

If the value returned to the file pointer in NULL, specifies file can't be created or opened.

If the character read from the file is EOF, any attempt to read beyond EOF creates an error.

feof() function marks the end of the file

Text File	Binary File
A **text file** stores data in the form of human readable format such as alphabets, digits and other special symbols by storing their ASCII values. For example, any file with a .txt, .rtf, etc as extension.	A **binary file** stores data as a sequence of bytes which are not in a human readable format. For example, files with .exe, .mp3,.png etc as extension.
An error in a **textual file** can be recognized and eliminated easily.	Whereas, an error in a **binary file** corrupts the file and is not easy to detect.
In the **text mode**, a special character with the ASCII code 26 is inserted at the end of the file. This character when encountered returns the EOF signal to the program.	In the **binary mode**, there is no any special character to signify the EOF. It keeps track with the help of the number of characters present in the directory entry of the file.
A text file Can store only plain text.	A binary file Can store different types of data (image, audio, text).
The bits in text files represent characters.	The bits in binary files represent custom data.