Unit 9. Data files

- Introduction to data files
- Types of files (text file, binary file)
- File handling operation
- Opening and closing file
- Creating file
- Library functions for READING from a file and WRITING to a file: (fputs, fgets, fputc, fgetc fprintf, fscanf)

File:

- A file is a container in computer storage devices used for storing data.
- A file represents a sequence of bytes, regardless of it being a text file or a binary file.
- C programming language provides access on high level functions as well as low level (OS level) calls to handle file on storage devices.

Why files are needed?

- To store data permanently.
- To organize data.
- To share data between users or between computers.

Types of Files

1. Text files

- Text files are the normal .txt files. We can easily create text files using any simple text editors such as Notepad. We can easily edit or delete the contents of the text file.
- They take minimum effort to maintain, are easily readable.

2. Binary files

- Binary files are mostly the .bin files in the computer.
- Instead of storing data in plain text, they store it in the binary form (0's and 1's).
- They can hold a higher amount of data, are not readable easily.

Working with files

- When working with files, we need to declare a pointer of type file. This declaration is needed for communication between the file and the program.

```
FILE *fptr;
```

File Operations

In C, you can perform four major operations on files, either text or binary:

1. Creating a new file
2. Opening an existing file
3. Closing a file
4. Reading from and writing information to a file

Opening a file- for creation and edit

- Opening a file is performed using the fopen() function.
- Example:

```
ptr = fopen("filepath","mode");
```

For example,

```
fopen("E:\\cprogram\\newprogram.txt","w");
```

- If newprogram.txt doesn't exist in the location E:\cprogram. The first function creates a new file named newprogram.txt.

Opening Modes in Standard I/O		
Mode	Meaning of Mode	During Inexistence of file
`r`	Open for reading.	If the file does not exist, `fopen()` returns NULL.
`w`	Open for writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
`a`	Open for append.	Data is added to the end of the file. If the file does not exist, it will be created.
`r+`	Open for both reading and writing.	If the file does not exist, `fopen()` returns NULL.
`w+`	Open for both reading and writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
`a+`	Open for both reading and appending.	If the file does not exist, it will be created.

Closing a File

- The file (both text and binary) should be closed after reading/writing.
- Closing a file is performed using the fclose() function.

```
fclose(fptr);
```

Here, fptr is a file pointer associated with the file to be closed.

Reading and writing to a text file

1. fputs() and fgets()

- fputs and fgets are used to read and write strings to files. fputs writes a string to a file, and fgets reads a string from a file.

Example:

```c
#include <stdio.h>

int main() {
  FILE *fp;
  char str[100];

  fp = fopen("myfile.txt", "w");
  fputs("Hello everyone, good morning\n", fp);
  fclose(fp);

  fp = fopen("myfile.txt", "r");
  fgets(str, 100, fp);
  printf("%s", str);
  fclose(fp);

  return 0;
}
```

2. fputc() and fgetc()

- fputc and fgetc are used to read and write single characters to files. fputc writes a character to a file, and fgetc reads a character from a file.

Example:

```c
#include <stdio.h>

int main()
{
    FILE *fp;
    char ch;

    fp = fopen("myfile.txt", "w");
    ch = 'A';
    fputc(ch, fp);
    fclose(fp);

    fp = fopen("myfile.txt", "r");
    ch = fgetc(fp);
    printf("%c", ch);
    fclose(fp);

    return 0;
}
```

3. fprintf() and fscanf()

- fprintf and fscanf are used to format and read data from files. fprintf writes formatted data to a file, and fscanf reads formatted data from a file.

Example:

```c
#include <stdio.h>

int main()
{
    FILE *fp;
    int age = 25;
    char name[] = "John Doe";

    fp = fopen("myfile.txt", "w");

    // Write the age to the file
    fprintf(fp, "%d\n", age);

    // Write the name to the file
    fprintf(fp, "%s\n", name);

    fclose(fp);

    fp = fopen("myfile.txt", "r");
```

```
    // Read the age from the file
    fscanf(fp, "%d", &age);

    // Read the name from the file
    fscanf(fp, "%s", name);

    printf("Age: %d\nName: %s\n", age, name);

    fclose(fp);

    return 0;
}
```

More Programs

Example 1: Write to a text file

```
#include <stdio.h>
int main()
{
   char message[100];
   FILE *fptr;

   fptr = fopen("C:\\Users\\Madan\\Desktop\\message.txt","w");

   if(fptr==NULL) printf("error");
   printf("Enter your message: ");
   gets(message);

   fprintf(fptr,message);
   fclose(fptr);

   return 0;
}
```

- This program takes a number from the user and stores in the file program.txt.

Example 2: Read from a text file

```
#include <stdio.h>
int main()
{
    char message[100];
    FILE *fptr;

    fptr = fopen("C:\\Users\\Madan\\Desktop\\message.txt","r");
```

```
    fgets(message, 100, fptr);
    printf("The message is: %s", message);
    fclose(fptr);

    return 0;
}
```

This program reads the integer present in the program.txt file and prints it onto the screen.

WACP to read name and marks of n number of students and store them in a file.

```
#include <stdio.h>
int main()
{
   char name[50];
   int marks, i, num;

   printf("Enter number of students: ");
   scanf("%d", &num);

   FILE *fptr;
   fptr = (fopen("C:\\student.txt", "w"));
   if(fptr == NULL)
   {
       printf("Error!");
       exit(1);
   }

   for(i = 0; i < num; ++i)
   {
      printf("For student%d\nEnter name: ", i+1);
      scanf("%s", name);

      printf("Enter marks: ");
      scanf("%d", &marks);

      fprintf(fptr,"\nName: %s \nMarks=%d \n", name, marks);
   }

   fclose(fptr);
   return 0;
```

```
}
```