FILE HANDLING

RESEARCH :

File handling in C is the process in which we create, open, read, write, and close operations on a file. C language provides different functions such as fopen(), fwrite(), fseek(), fprintf(), etc. to perform input, output, and many different C file operations in our program.
C File Operations

C file operations refer to the different possible operations that we can perform on a file in C such as:

1. Creating a new file – fopen() with attributes as "a" or "a+" or "w" or "w+"
2. Opening an existing file – fopen()
3. Reading from file – fscanf() or fgets()
4. Writing to a file – fprintf() or fputs()
5. Moving to a specific location in a file – fseek(), rewind()
6. Closing a file – fclose()

SOURCE :

1. **File Handling in C - GeeksforGeeks** and fpl notes File Handling

In C, you can create, open, read, and write to files by declaring a pointer of type FILE, and use the fopen() function:

```
FILE *fptr;

fptr = fopen(filename,mode);
```

FILE is basically a data type, and we need to create a pointer variable to work with it (fptr). For now, this line is not important. It's just something you need when working with files.

To actually open a file, use the fopen() function, which takes two parameters:

Parameter	Description
FILENAME	The name of the actual file you want to open (or create), like filename.txt
MODE	A single character, which represents what you want to do with the file (read, write or append): w - Writes to a file a - Appends new data to a file r - Reads from a file

2. C Files - File Handling and How To Create Files

ANALYSIS :

Over analysing the research part we can conclude that the file handling can be used for various purposes like browsing the student’s data files, company’s employee detail files and many more. File handling in C is the process of creating, opening, reading, writing, and closing files to store data permanently. It uses a FILE pointer and functions like fopen(), fclose(), fprintf(), fscanf(), fgets(), fputs(), fread(), and fwrite(). Different modes (r, w, a, r+, w+, a+) control whether you read, overwrite, or append data. Text files store human-readable data, while binary files store structured records more efficiently. The workflow is always: declare a FILE*, open the file, perform operations, and close it. File handling is widely used in student records, digital libraries, log systems, and configuration files. It ensures persistence of data beyond program execution.

IDEATE :

Here we are going to create a c program where we will write the students name,roll no. and marksheet and then showing it for reading.

ALGORITHM :

1. Start: Initialize the program.

2. Declare variables**:** DefineFILE *fp, char name[50], and int roll, marks .

3. Open file (write mode): Use fopen("marksheet.txt", "w"); if fp == NULL, print error and exit.

4. Input data: Read name with fgets(), then roll and marks with scanf().

5. Use fprintf() to write name, roll number, and marks to the file.

6. Close file (after write)**:** Call fclose(fp) and print success confirmation.

7. Open file (read mode): Use fopen("marksheet.txt", "r"); if fp == NULL, print error and exit.

8. Display contents: Read characters using fgetc() in a loop until EOF.

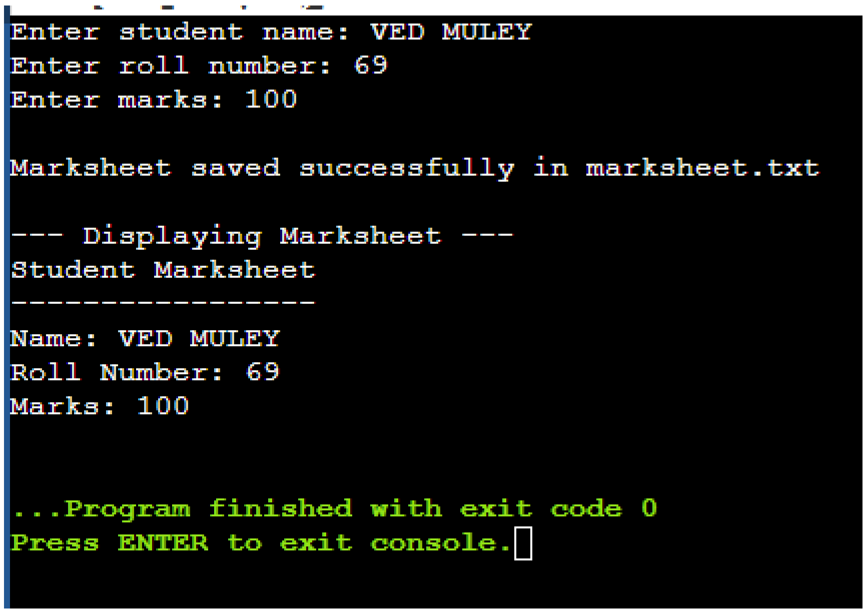
9. Close file (after read): Call fclose(fp).

10. End: Return from main() and terminate the program.

BUILD :

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    FILE *fp;
    char name[50];
    int roll, marks;

    fp = fopen("marksheet.txt", "w");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    printf("Enter student name: ");
    fgets(name, sizeof(name), stdin);

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

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

    fprintf(fp, "Student Marksheet\n");
    fprintf(fp, "-----------------\n");
    fprintf(fp, "Name: %s", name);
    fprintf(fp, "Roll Number: %d\n", roll);
    fprintf(fp, "Marks: %d\n", marks);

    fclose(fp);
    printf("\nMarksheet saved successfully in marksheet.txt\n");
    fp = fopen("marksheet.txt", "r");
    if (fp == NULL) {
        printf("Error opening file for reading!\n");
        return 1;
    }

    char ch;
    printf("\n--- Displaying Marksheet ---\n");
    while ((ch = fgetc(fp)) != EOF) {
        putchar(ch);
    }
    fclose(fp);

    return 0;
}
```

OUTPUT :



```
Enter student name: VED MULEY
Enter roll number: 69
Enter marks: 100

Marksheet saved successfully in marksheet.txt

--- Displaying Marksheet ---
Student Marksheet
-----------------
Name: VED MULEY
Roll Number: 69
Marks: 100


...Program finished with exit code 0
Press ENTER to exit console.
```

IMPLEMENTATION :

https://github.com/vedmuley536/Ved-Muley