Date: Lab Report No.: **10** Set: **B**

TITLE OF THE PROGRAM: **File Handling**

Objective

The objective of these programs is to demonstrate various file operations in C, including storing and retrieving student details using structures and file handling functions like **fopen, fwrite**, **fread**, and standard I/O operations. These programs cover:

- Storing student details until the user decides to stop.
- Using structures to store and retrieve student details.
- Working with multiple student records.
- File operations such as writing to and reading from files.
- Basic file manipulations like deleting and renaming files.

Requirement

1. C Compiler (e.g., GCC)
2. Computer System
3. IDE or Text Editor
4. OS compatible with the software

THEORY

File handling in C involves interacting with files through operations such as opening, reading, writing, closing, deleting, and renaming. The FILE pointer is used to manage file operations, and various functions like **fopen(), fprintf(), fread(), fwrite(), fclose(), remove(), rename()** are employed to perform these operations.

Structures in C are user-defined data types that allow the grouping of variables of different types under a single name. Using structures with file handling enables efficient storage and retrieval of complex data, such as student records or employee details.

Procedure

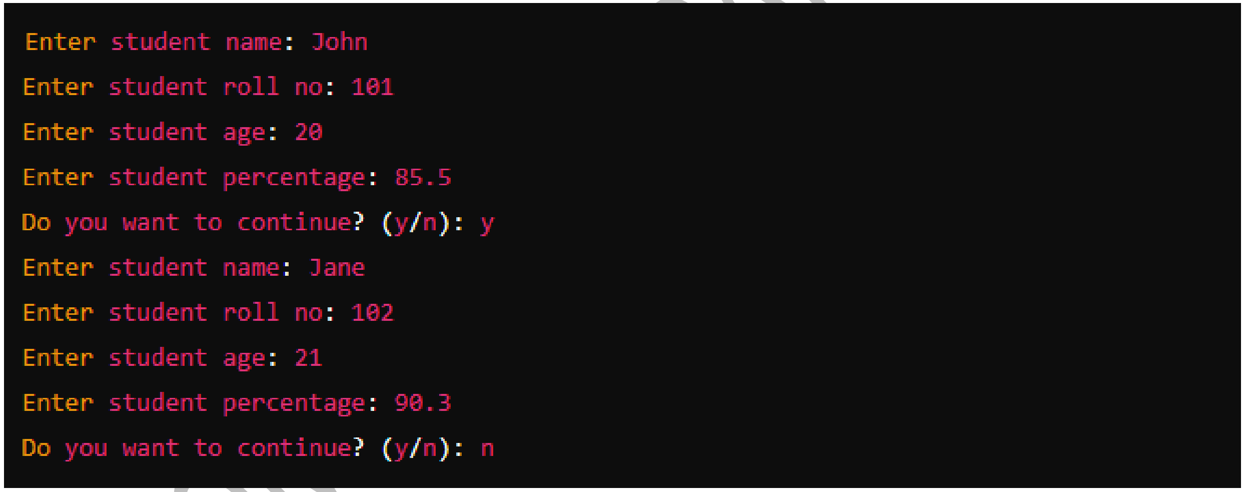
1. **Storing Details of Students until User Presses 'n' or Any Other Character than 'y'**

```
#include<stdio.h>

        int main ()
```

```
{

        char name[50]; // Array to store the name of the student
        int rollno;    // Variable to store the roll number of the student
        int age;       // Variable to store the age of the student
        float per;     // Variable to store the percentage of the student
        char choice;   // Variable to store the user's choice to continue or not
        FILE *fp;      // File pointer to handle file operations

        // Open file in write mode
        fp = fopen("student123.txt", "w");

        do {
          // Taking student details as input
          printf("Enter student name: ");
          scanf("%s", name);

          printf("Enter student roll no: ");
          scanf("%d", &rollno);

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

          printf("Enter student percentage: ");
          scanf("%f", &per);

          // Writing the student details to the file
          fprintf(fp, "%s\t%d\t%d\t%f\n", name, rollno, age, per);

          // Asking the user if they want to continue entering more data
```

```
                printf("Do you want to continue? (y/n): ");
                choice = getche();

            } while (choice == 'Y' || choice == 'y'); // Continue if the user enters 'y' or 'Y'

            // Close the file
            fclose(fp);

            return 0;
}
```

Output



```
Enter student name: John
Enter student roll no: 101
Enter student age: 20
Enter student percentage: 85.5
Do you want to continue? (y/n): y
Enter student name: Jane
Enter student roll no: 102
Enter student age: 21
Enter student percentage: 90.3
Do you want to continue? (y/n): n
```

Explanation:

- This program stores student details in a file named student123.txt.
- The user is prompted to input data until they choose to stop.
- The file is closed once data entry is complete.

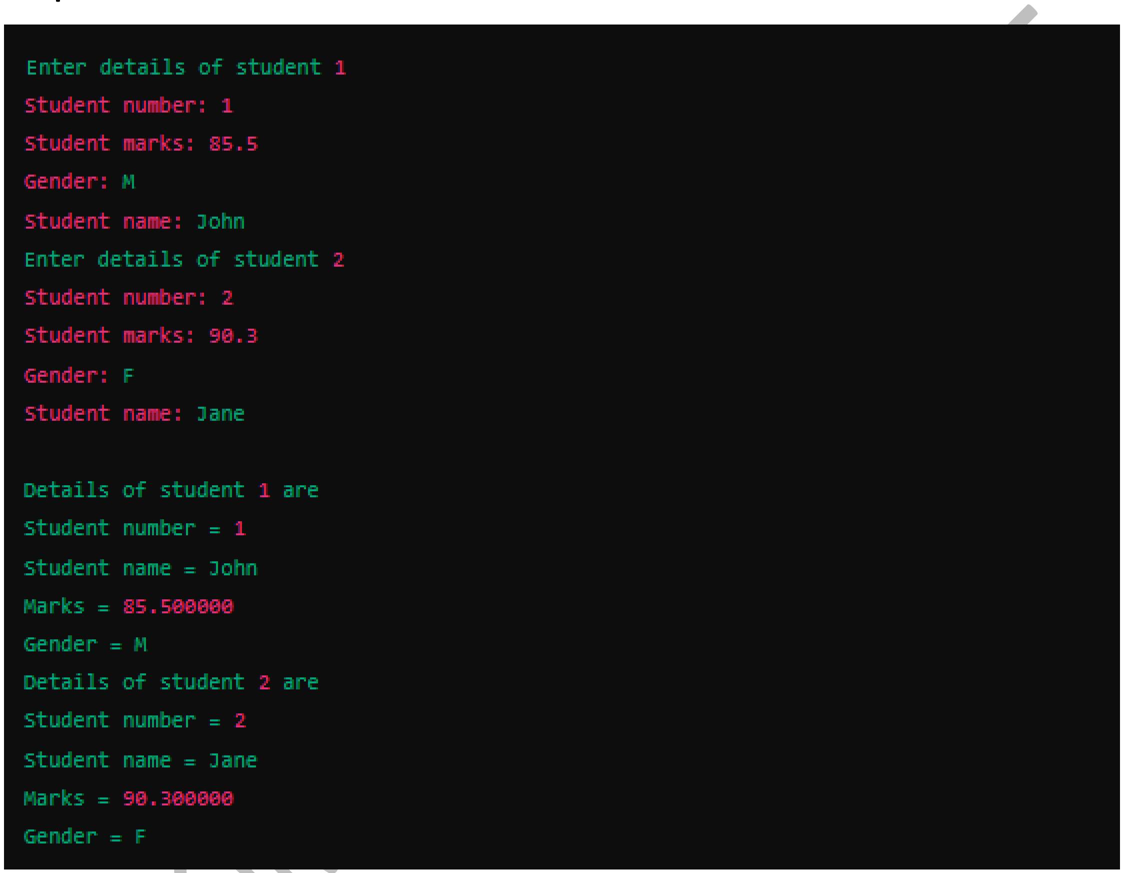
2. **Storing Details of 2 students Using Structure and File Operations**

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

```
// Define a structure to store student details
struct student {
   int sno;       // Student number
   char sname[30]; // Student name
   float marks;    // Student marks
   char gender;    // Student gender
};

int main () {
   struct student s[60]; // Array of structures to store multiple student details
   int i;               // Loop variable
   FILE *fp;            // File pointer to handle file operations

   // Open file in write mode
   fp = fopen("student1.txt", "w");

   // Loop to take input for 2 students
   for (i = 0; i < 2; i++) {
      printf("Enter details of student %d\n", i + 1);

      printf("Student number: ");
      scanf("%d", &s[i].sno);

      printf("Student marks: ");
      scanf("%f", &s[i].marks);

      printf("Gender: ");
      fflush(stdin);
```

```
        scanf("%c", &s[i].gender);

        printf("Student name: ");
        fflush(stdin);
        gets(s[i].sname);

        // Write the student details to the file
        fwrite(&s[i], sizeof(s[i]), 1, fp);
    }

    // Close the file
    fclose(fp);

    // Open file in read mode
    fp = fopen("student1.txt", "r");

    // Loop to read and display student details from the file
    for (i = 0; i < 2; i++) {
        printf("Details of student %d are\n", i + 1);

        fread(&s[i], sizeof(s[i]), 1, fp);

        printf("Student number = %d\n", s[i].sno);
        printf("Student name = %s\n", s[i].sname);
        printf("Marks = %f\n", s[i].marks);
        printf("Gender = %c\n", s[i].gender);
    }

    // Close the file
    fclose(fp);

    return 0;
}
```

Output



```
Enter details of student 1
Student number: 1
Student marks: 85.5
Gender: M
Student name: John
Enter details of student 2
Student number: 2
Student marks: 90.3
Gender: F
Student name: Jane

Details of student 1 are
Student number = 1
Student name = John
Marks = 85.500000
Gender = M
Details of student 2 are
Student number = 2
Student name = Jane
Marks = 90.300000
Gender = F
```

Explanation:

- This program uses a structure to store and retrieve student details.
- The data is written to a file using fwrite() and read back using fread().
- The file is reopened in read mode to display the stored records.

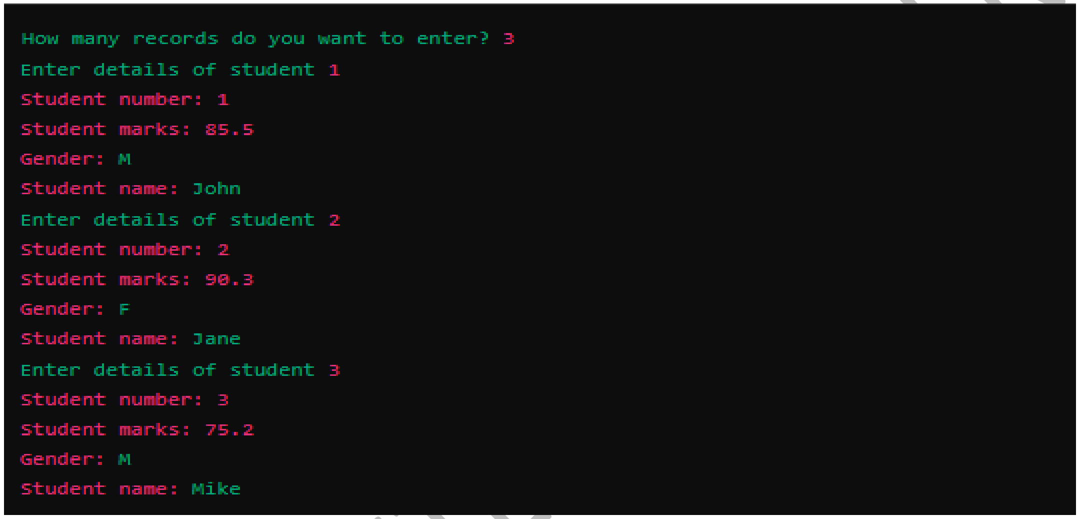
3. **Storing Details of N Students Using Structure and File Operations**

```
#include<stdio.h>

// Define a structure to store student details
```

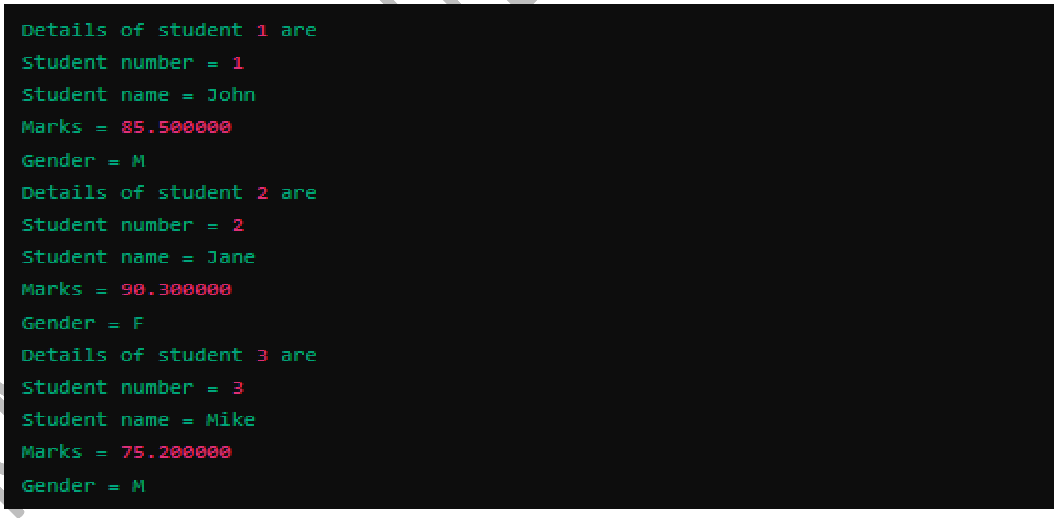
```
struct student {
   int sno;       // Student number
   char sname[30]; // Student name
   float marks;    // Student marks
   char gender;    // Student gender
};

int main () {
   struct student s[60]; // Array of structures to store multiple student details
   int i, n;           // Loop variable and variable to store number of students
   FILE *fp;           // File pointer to handle file operations

   // Open file in write mode
   fp = fopen("student1.txt", "w");

   // Prompt user to enter the number of students
   printf("How many records do you want to enter? ");
   scanf("%d", &n);

   // Loop to take input for n students
   for (i = 0; i < n; i++) {
      printf("Enter details of student %d\n", i + 1);

      printf("Student number: ");
      scanf("%d", &s[i].sno);

      printf("Student marks: ");
      scanf("%f", &s[i].marks);
```

```
        printf("Gender: ");
        fflush(stdin);
        scanf("%c", &s[i].gender);

        printf("Student name: ");
        fflush(stdin);
        gets(s[i].sname);

        // Write the student details to the file
        fwrite(&s[i], sizeof(s[i]), 1, fp);
    }

    // Close the file
    fclose(fp);

    // Open file in read mode
    fp = fopen("student1.txt", "r");

    // Loop to read and display student details from the file
    for (i = 0; i < n; i++) {
        printf("Details of student %d are\n", i + 1);

        fread(&s[i], sizeof(s[i]), 1, fp);

        printf("Student number = %d\n", s[i].sno);
        printf("Student name = %s\n", s[i].sname);
        printf("Marks = %f\n", s[i].marks);
        printf("Gender = %c\n", s[i].gender);
    }
    // Close the file
    fclose(fp);

    return 0;
}
```

Output



```
How many records do you want to enter? 3
Enter details of student 1
Student number: 1
Student marks: 85.5
Gender: M
Student name: John
Enter details of student 2
Student number: 2
Student marks: 90.3
Gender: F
Student name: Jane
Enter details of student 3
Student number: 3
Student marks: 75.2
Gender: M
Student name: Mike
```



```
Details of student 1 are
Student number = 1
Student name = John
Marks = 85.500000
Gender = M
Details of student 2 are
Student number = 2
Student name = Jane
Marks = 90.300000
Gender = F
Details of student 3 are
Student number = 3
Student name = Mike
Marks = 75.200000
Gender = M
```

Explanation:

- This program is similar to the previous one but allows the user to specify how many student records to enter.
- The data is stored and retrieved using structures and file operations.

4. **Storing Details of 10 Employees**

```
#include<stdio.h>
int main() {
   char name[20], add[30]; // Arrays to store name and address of the employee
   int salary, i;         // Variables to store salary and loop variable
   FILE *fp;              // File pointer to handle file operations

   // Open file in write mode
   fp = fopen("record.txt", "w");

   // Loop to take input for 10 employees
   for (i = 0; i < 10; i++) {
      printf("Input name, address, and salary of an employee:\n");
      scanf("%s%s%d", name, add, &salary);
      // Write the employee details to the file
      fprintf(fp, "%s %s %d\n", name, add, salary);
   }

   // Close the file
   fclose(fp);

   return 0;
}
```

Output

```
Input name, address, and salary of an employee:
John NewYork 50000
Input name, address, and salary of an employee:
Jane LosAngeles 55000
...
```

Explanation:

This program stores the details of 10 employees in a file **record.txt**.

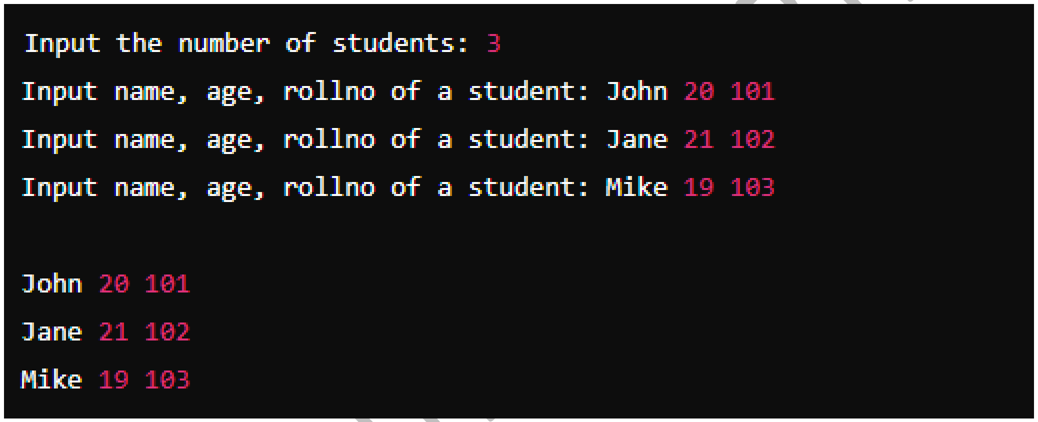
The file includes the employee's name, address, and salary.

5. **Creating and Displaying Student Details**

```
 #include<stdio.h>
int main() {
   char name[20]; // Array to store the name of the student
   int age, rollno, n, i; // Variables to store age, roll number, and loop variables
   FILE *fp; // File pointer to handle file operations
   // Prompt user to enter the number of students
   printf("Input the number of students: ");
   scanf("%d", &n);
   // Open file in write mode
   fp = fopen("std.txt", "w");
   // Loop to take input for n students
   for (i = 0; i < n; i++) {
      printf("Input name, age, rollno of a student: ");
      scanf("%s%d%d", name, &age, &rollno);
      // Write the student details to the file
      fprintf(fp, "%s %d %d\n", name, age, rollno);
   }

   // Close the file
   fclose(fp);

   // Open file in read mode
   fp = fopen("std.txt", "r");
```

```
    // Loop to read and display student details from the file
    while (fscanf(fp, "%s%d%d", name, &age, &rollno) != EOF) {
      printf("\n%s %d %d\n", name, age, rollno);
    }
  // Close the file
 fclose(fp);
  return 0;
}
```

Output



```
Input the number of students: 3
Input name, age, rollno of a student: John 20 101
Input name, age, rollno of a student: Jane 21 102
Input name, age, rollno of a student: Mike 19 103

John 20 101
Jane 21 102
Mike 19 103
```

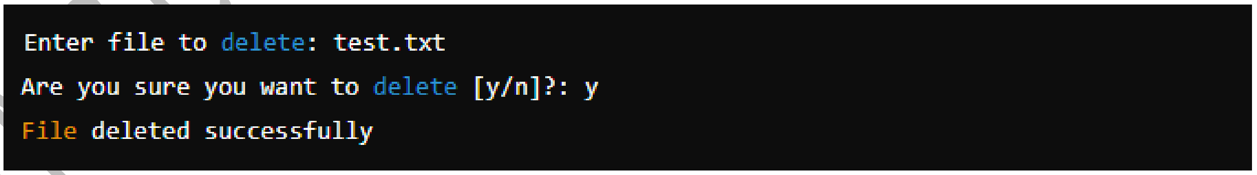
Explanation:

- This program creates a file to store student records and then displays the stored records.
- The user specifies the number of students, and the details are written to std.txt.

6. **Deleting a File**

```
#include<stdio.h>
int main()
{
 char file[30], ans; // Array to store the file name and variable to store user's confirmation


    // Prompt user to enter the file name to delete
    printf("Enter file to delete: ");
    gets(file);

    // Prompt user for confirmation
    printf("Are you sure you want to delete [y/n]?: ");
    ans = getchar();

    // Check user's confirmation
    if (ans == 'y' || ans == 'Y') {
      // Try to delete the file
      if (remove(file) == 0)
        printf("File deleted successfully\n");
      else
        printf("Error deleting file! Try again\n");
    } else {
      printf("Delete process is cancelled\n");
    }

    return 0;
}
```

Output



```
Enter file to delete: test.txt
Are you sure you want to delete [y/n]?: y
File deleted successfully
```

Explanation:

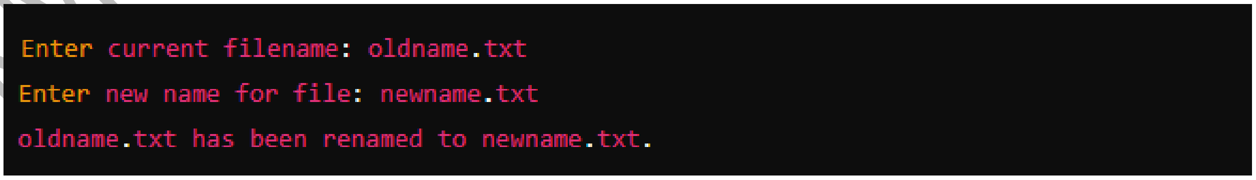
- This program prompts the user to delete a file.
- If the user confirms, the file is deleted using the remove() function.

7. **Renaming a File**

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

```c
int main()
 {
   char oldname[30], newname[30]; // Arrays to store the old and new file names

   // Prompt user to enter the current file name
   printf("Enter current filename: ");
   gets(oldname);

   // Prompt user to enter the new file name
   printf("Enter new name for file: ");
   gets(newname);

   // Try to rename the file
   if (rename(oldname, newname) == 0) {
      printf("%s has been renamed to %s.\n", oldname, newname);
   } else {
      printf("An error has occurred renaming %s.\n", oldname);
   }

   return 0;
}
```

Output



```
Enter current filename: oldname.txt
Enter new name for file: newname.txt
oldname.txt has been renamed to newname.txt.
```

Conclusion

In conclusion, the various programs demonstrated the fundamental concepts of file handling in C, such as storing and retrieving data using structures, handling multiple records, and performing basic file manipulations like deleting and renaming files. By leveraging functions like **fopen, fwrite, fread, fprintf**, and file pointers, we efficiently stored complex data, such as student and employee records, in files. The exercises also highlighted the importance of file handling in managing persistent data storage, showcasing how data can be manipulated and accessed across different runs of a program. Through these tasks, we developed a deeper understanding of how to implement and manage data files in C, an essential skill for creating reliable and efficient software applications.