Name: SATYAM PANDEY
FE – B1 AIDS
2401074

Experiment No. 1

Aim: Write a program to perform file handling operations

Theory:

1. File Creation and Writing:
    - Opens the file in write mode ("w").
    - Writes content to the file using fprintf.
2. File Reading:
    - Opens the file in read mode ("r").
    - Reads and displays the content using fgetc.
3. File Appending:
    - Opens the file in append mode ("a").
    - Adds new content without overwriting existing data.
4. File Deletion:
    - Deletes the file using the remove function.

Compilation and Execution:

1. Save the code to a file, e.g., file_handling.c.
2. Compile using gcc file_handling.c -o file_handling.
3. Run the program with ./file_handling.

Program:

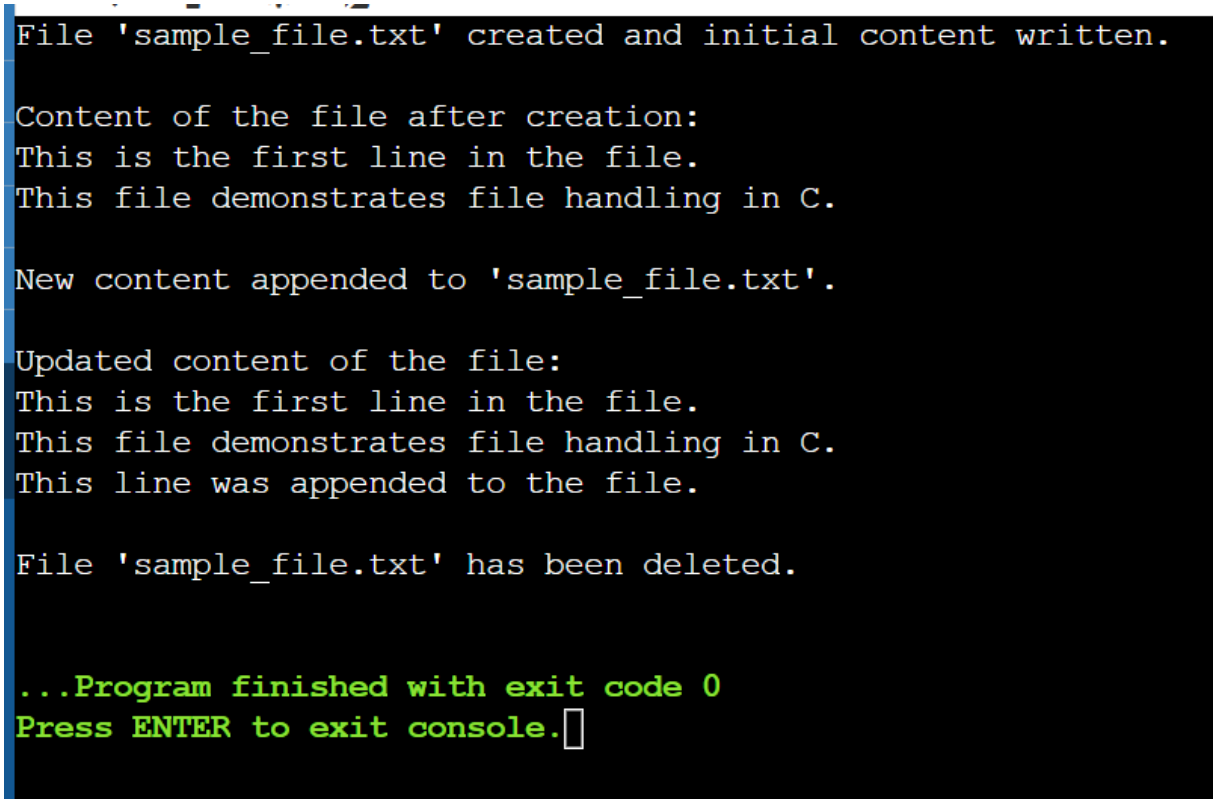
```
#include <stdio.h>
#include <stdlib.h>

void file_handling_operations() {
   char filename[] = "sample_file.txt";
   FILE *file;

   file = fopen(filename, "w");
   if (file == NULL) {
      printf("Error opening file for writing!\n");
      exit(1);
   }
   fprintf(file, "This is the first line in the file.\n");
   fprintf(file, "This file demonstrates file handling in C.\n");
   fclose(file);
   printf("File '%s' created and initial content written.\n", filename);

   file = fopen(filename, "r");
   if (file == NULL) {
      printf("Error opening file for reading!\n");
      exit(1);
   }
   printf("\nContent of the file after creation:\n");
   char ch;
   while ((ch = fgetc(file)) != EOF) {
      putchar(ch);
   }
```

Name: SATYAM PANDEY
FE – B1 AIDS
2401074

```
    fclose(file);

    file = fopen(filename, "a");
    if (file == NULL) {
        printf("Error opening file for appending!\n");
        exit(1);
    }
    fprintf(file, "This line was appended to the file.\n");
    fclose(file);
    printf("\nNew content appended to '%s'.\n", filename);

    file = fopen(filename, "r");
    if (file == NULL) {
        printf("Error opening file for reading!\n");
        exit(1);
    }
    printf("\nUpdated content of the file:\n");
    while ((ch = fgetc(file)) != EOF) {
        putchar(ch);
    }
    fclose(file);

    if (remove(filename) == 0) {
        printf("\nFile '%s' has been deleted.\n", filename);
    } else {
        printf("\nError deleting the file '%s'.\n", filename);
    }
}

int main() {
    file_handling_operations();
    return 0;
}
```

Name: SATYAM PANDEY
FE – B1 AIDS
2401074

Output:



Conclusion:

File handling is an essential feature of programming that allows you to create, read, write, append, and delete files. It provides a way to store and manage data persistently, beyond the program's runtime.

Key Takeaways:

1. **Purpose**:
   - File handling enables long-term data storage and retrieval.
   - It's crucial for applications requiring logs, configuration files, or data persistence.
2. **Operations**:
   - **Creation**: Files can be created to hold data.
   - **Reading**: Allows retrieving stored data for use in programs.
   - **Writing**: Enables overwriting or adding new data to files.
   - **Appending**: Adds new data without affecting existing content.
   - **Deletion**: Removes files when no longer needed.
3. **Language-Specific Implementation**:
   - Different programming languages offer unique APIs for file handling, as demonstrated in Python and C examples.
   - Python simplifies file handling with high-level abstractions.
   - C provides low-level control over file operations using standard I/O functions.
4. **Practical Use Cases**:
   - Logs for debugging or system operations.
   - Configuration and settings storage.
   - Data export/import in various formats (e.g., CSV, JSON).
   - Persistent data storage for applications.

Name: SATYAM PANDEY
FE – B1 AIDS
2401074