Trainer: Asif Nafees

File Handling

File handling is an essential part of C programming that allows you to store data permanently on a disk rather than in memory. It involves working with files and performing operations like reading, writing, and modifying files.

Types of File Operations
1. Creating a file
2. Opening a file
3. Reading from a file
4. Writing to a file
5. Closing a file

File Pointers
In C, a file is managed using a FILE pointer. The FILE pointer points to a structure that holds information about the file.
FILE *filePtr;

Basic File Handling Functions
1. fopen(): Opens a file.
2. fclose(): Closes a file.
3. fprintf(): Writes formatted data to a file.
4. fscanf(): Reads formatted data from a file.
5. fgetc(): Reads a single character from a file.
6. fputc(): Writes a single character to a file.
7. fgets(): Reads a string from a file.
8. fputs(): Writes a string to a file.
9. fseek(): Moves the file pointer to a specific location.
10. ftell(): Returns the current position of the file pointer.
11. rewind(): Moves the file pointer to the beginning of the file.

Example 1: Writing to a File
This program writes user input to a text file.

```
#include <stdio.h>
int main() {
    FILE *filePtr;
    char data[100];
    // Open the file in write mode
    filePtr = fopen("example.txt", "w");
    // Check if the file opened successfully
    if (filePtr == NULL) {
        printf("Error opening file.\n");
```

```
        return 1;
    }
    printf("Enter data to write to the file: ");
    fgets(data, sizeof(data), stdin);
    // Write data to the file
    fprintf(filePtr, "%s", data);
    printf("Data written successfully.\n");
    // Close the file
    fclose(filePtr);
    return 0;
}
```

Output:
Enter data to write to the file: Hello, File Handling in C!
Data written successfully.
The content of example.txt will be:
Hello, File Handling in C!

Example 2: Reading from a File
This program reads the content of a file and displays it on the screen.

```
#include <stdio.h>
int main() {
    FILE *filePtr;
    char ch;
    // Open the file in read mode
    filePtr = fopen("example.txt", "r");
    // Check if the file opened successfully
    if (filePtr == NULL) {
        printf("Error opening file.\n");
        return 1;
    }
    printf("File Content:\n");
    // Read and display file content character by character
    while ((ch = fgetc(filePtr)) != EOF) {
        putchar(ch);
    }
    // Close the file
    fclose(filePtr);
    return 0;
}
```

Output:
File Content:
Hello, File Handling in C!

Example 3: Appending to a File
This program appends data to an existing file.

```
#include <stdio.h>
int main() {
    FILE *filePtr;
    char data[100];
    // Open the file in append mode
    filePtr = fopen("example.txt", "a");
    // Check if the file opened successfully
    if (filePtr == NULL) {
        printf("Error opening file.\n");
        return 1;
    }
    printf("Enter data to append to the file: ");
    fgets(data, sizeof(data), stdin);
    // Append data to the file
    fputs(data, filePtr);
    printf("Data appended successfully.\n");
    // Close the file
    fclose(filePtr);
    return 0;
}
```

Output:

```
Enter data to append to the file: Appending new text!
Data appended successfully.
The content of example.txt after appending will be:
Hello, File Handling in C!
Appending new text!
```

Example 4: Using fseek(), ftell(), and rewind()
This program demonstrates moving the file pointer.

```
#include <stdio.h>
int main() {
    FILE *filePtr;
    long position;
    // Open the file in read mode
    filePtr = fopen("example.txt", "r");
    if (filePtr == NULL) {
        printf("Error opening file.\n");
        return 1;
    }
```

```
    // Move the file pointer to the 5th character
    fseek(filePtr, 5, SEEK_SET);
    printf("File pointer moved to the 5th character.\n");
    // Get the current position of the file pointer
    position = ftell(filePtr);
    printf("Current position: %ld\n", position);
    // Read and display the next character
    printf("Character at current position: %c\n", fgetc(filePtr));
    // Rewind the file pointer to the beginning
    rewind(filePtr);
    printf("File pointer rewound to the beginning.\n");
    // Close the file
    fclose(filePtr);
    return 0;
}
```

Output:
File pointer moved to the 5th character.
Current position: 5
Character at current position: F
File pointer rewound to the beginning.