Unit 7: Input Output and File Handling

File handling in C allows us to store data permanently in files instead of temporary memory (RAM).

7.1 Concept of File Handling

◇ Definition:

File handling is used to:

- Store data permanently
- Read data from file
- Write data into file
- Modify file content

◇ Why File Handling?

- Data is not lost after program ends
- Used for large data storage
- Useful in real applications (bank, student records, etc.)

7.2 File Access Methods

There are two main file access methods:

1 Sequential Access

- Data is accessed one after another.
- Cannot jump directly to middle.
- Example: reading file line by line.

2 Random Access (Direct Access)

- Can access any location directly.
- Uses `fseek()` function.

- Faster for large files.

7.3 Functions of File Handling

All file handling functions are declared in:

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

Files are handled using pointer:

```
FILE *fp;
```

◇ fopen()

Opens a file.

Syntax:

```
fp = fopen("filename", "mode");
```

Modes:

Mode	Meaning
r	Read
w	Write (creates new file)
a	Append
r+	Read & Write
w+	Read & Write (overwrite)
a+	Read & Append

◇ fclose()

Closes the file.

```
fclose(fp);
```

◇ fflush()

Clears buffer and writes output immediately.

```
fflush(fp);
```

◇ freopen()

Redirects file stream.

```
freopen("file.txt", "w", stdout);
```

Example Program (Write to File)

```
#include <stdio.h>

int main()
{
    FILE *fp;
    fp = fopen("data.txt", "w");

    fprintf(fp, "Hello File Handling");

    fclose(fp);

    return 0;
}
```

7.4 Formatted Input Output

Uses formatted functions:

- `fprintf()` → write formatted output
- `fscanf()` → read formatted input

Example

```
#include <stdio.h>

int main()
{
    FILE *fp;
    int num;

    fp = fopen("number.txt", "w");
    fprintf(fp, "%d", 100);
    fclose(fp);

    fp = fopen("number.txt", "r");
    fscanf(fp, "%d", &num);
    printf("Number = %d", num);
    fclose(fp);

    return 0;
}
```

7.5 Character Input Output

Used to read/write single characters.

- `fgetc()` → Read character
- `fputc()` → Write character

Example

```
#include <stdio.h>

int main()
```

```
{
    FILE *fp;
    char ch;

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

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

    return 0;
}
```

7.6 Direct Input Output

Used for binary files.

- `fread()`
- `fwrite()`

Example

```
#include <stdio.h>

struct Student
{
    int roll;
    float marks;
};

int main()
{
    FILE *fp;
    struct Student s = {1, 90.5};
```

```
    fp = fopen("student.dat", "wb");
    fwrite(&s, sizeof(s), 1, fp);
    fclose(fp);

    return 0;
}
```

7.7 Random File Access

Used to move file pointer.

Important Functions:

- `fseek()`
- `ftell()`
- `rewind()`

◇ fseek()

```
fseek(fp, offset, position);
```

Position:

- SEEK_SET
- SEEK_CUR
- SEEK_END

☑ Example

```
#include <stdio.h>

int main()
{
    FILE *fp;
    fp = fopen("data.txt", "r");
```

```
    fseek(fp, 5, SEEK_SET);
    printf("Position = %ld", ftell(fp));

    fclose(fp);
    return 0;
}
```

7.8 Error Handling

Check if file opened successfully:

```
if(fp == NULL)
{
    printf("Error opening file");
}
```

Other Error Functions:

- `feof(fp)` → Check end of file
- `ferror(fp)` → Check file error
- `perror("Error message")` → Print error

☑ Example

```
#include <stdio.h>

int main()
{
    FILE *fp;
    fp = fopen("data.txt", "r");

    if(fp == NULL)
    {
        printf("File not found!");
        return 1;
    }
```

```
    fclose(fp);
    return 0;
}
```

7.9 File Operations

Common operations:

Operation	Function
Open	fopen()
Close	fclose()
Read	fscanf(), fread(), fgetc()
Write	fprintf(), fwrite(), fputc()
Move pointer	fseek()
Check position	ftell()
Check EOF	feof()
Rename file	rename()
Delete file	remove()

☑ Example (Rename & Delete)

```
#include <stdio.h>

int main()
{
    rename("old.txt", "new.txt");
    remove("new.txt");
    return 0;
}
```

☑ Summary (Exam Ready)

Topic	Key Point

File handling	Permanent storage
fopen()	Open file
fclose()	Close file
fprintf()	Formatted write
fscanf()	Formatted read
fgetc()	Read character
fwrite()	Binary write
fseek()	Random access
Error handling	Check NULL