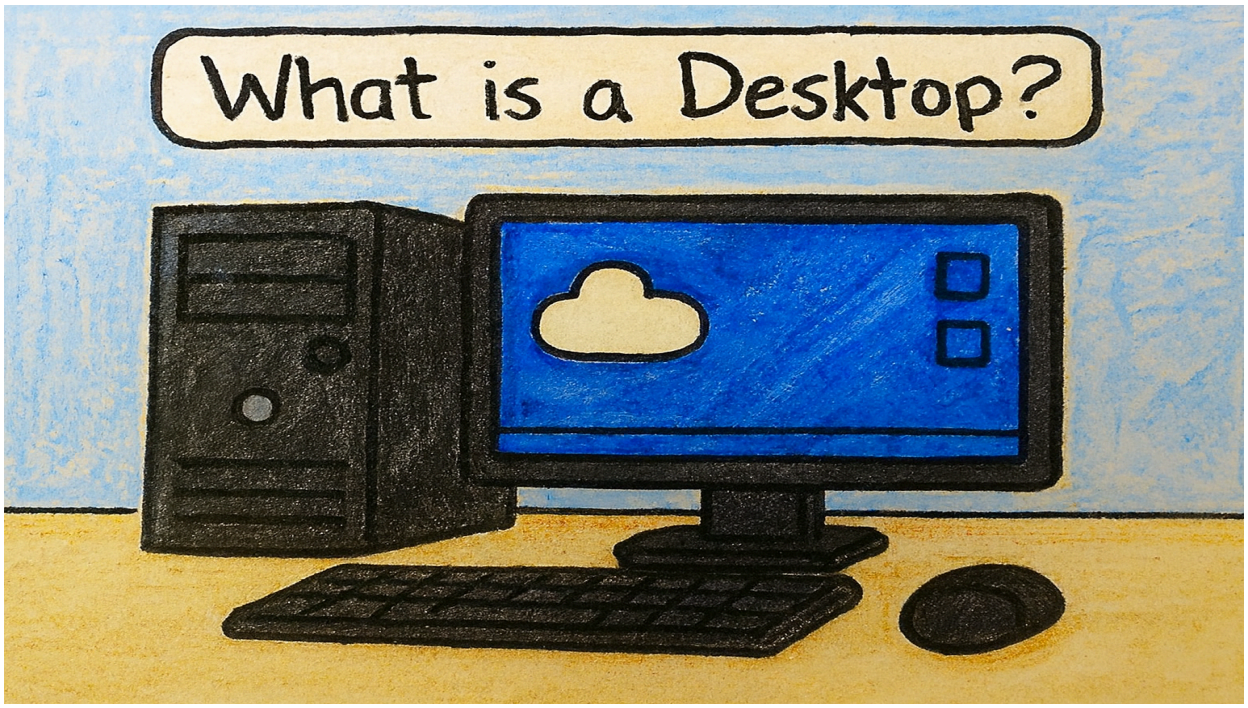

Class: 12th

Subject: Computer

Chapter 14: FILE HANDLING IN C

📍 Important MCQs:

1. Earlier C programs could not store data permanently because they worked with:

(a) dynamic data

(b) temporary data only ✅

(c) binary data

(d) external devices

2. Permanent data in C is stored in the form of:

(a) variables

(b) arrays

(c) files ✅

(d) pointers

3. A file is defined as:

(a) a collection of instructions

(b) a set of related records ✅

(c) a group of variables

(d) a memory location

4. File handling functions in C are provided by:

(a) stdlib.h

(b) string.h

(c) stdio.h ✅

(d) conio.h

5. A stream in C is:

(a) a physical device

(b) a data type

(c) a logical interface to a file ✅

(d) a memory block

6. A stream is associated with a file using:

(a) read operation

(b) write operation

(c) open operation ✅

(d) close operation

7. A stream is disassociated from a file using:

(a) fopen()

(b) fread()

(c) fclose() ✅

(d) fprintf()

8. A text stream is a sequence of:

(a) bytes

(b) records

(c) characters ✅

(d) integers

9. In a text stream, which of the following may occur?

(a) one-to-one byte correspondence

(b) character translation ✅

(c) no translation

(d) data padding

10. A binary stream is a sequence of:

(a) characters

(b) records

(c) lines

(d) bytes ✅

11. In a binary stream, data has:

(a) one-to-many correspondence

(b) many-to-one correspondence

(c) one-to-one correspondence ✅

(d) no correspondence

12. In C, a file can refer to:

(a) only disk files

(b) only keyboard input

(c) disk files, screen, keyboard, ports, etc. ✅

(d) only text files

13. The end of a text file in C is marked by:

(a) NULL

(b) \n

(c) EOF ✅

(d) TAB

14. Pressing the ENTER key while creating a text file inserts:

(a) EOF character

(b) space character

(c) newline character (\n) ✅

(d) tab character

15. Before reading from or writing to a file, it must be:

(a) compiled

(b) saved

(c) opened ✅

(d) closed

16. Which function is used to open a file in C?

(a) open()

(b) fileopen()

(c) fopen() ✅

(d) read()

17. The return type of fopen() is:

(a) int

(b) char*

(c) FILE* ✅

(d) void

18. If fopen() fails to open a file, it returns:

(a) 0

(b) EOF

(c) NULL ✅

(d) -1

19. Which mode opens an existing text file for reading only?

(a) w

(b) a

(c) r ✅

(d) r+

20. A file pointer in C is declared as:

(a) FILE fp;

(b) FILE &fp;

(c) FILE* fp; ✅

(d) pointer FILE fp;

21. When a program has no further use of a file, it should be:

(a) reopened

(b) deleted

(c) closed using fclose() ✅

(d) renamed

22. Which function is used to close a file in C?

(a) close()

(b) fileclose()

(c) fclose() ✅

(d) endfile()

23. The return type of the fclose() function is:

(a) void

(b) char

(c) int ✅

(d) FILE*

24. The fclose() function returns ____ on successful execution.

(a) EOF

(b) NULL

(c) 1

(d) 0 ✅

25. The fclose() function returns ____ if an error occurs.

(a) 0

(b) -1

(c) NULL

(d) EOF ✅

26. Which function is used to read a character from a file?

(a) readc()

(b) getchar()

(c) getc() ✅

(d) fscanf()

27. The getc() function returns EOF when:

(a) a character is read

(b) a file is opened

(c) end of file is reached or an error occurs ✅

(d) file is closed

28. Which function is used to write a character to a file?

(a) writec()

(b) putc() ✅

(c) putchar()

(d) fprintf()

29. The putc() function returns ____ if successful.

(a) EOF

(b) NULL

(c) the character written ✅

(d) 0

30. A string in C is stored as:

(a) a pointer

(b) a structure

(c) an array of characters ✅

(d) a single variable

31. An array is defined as:

(a) random memory locations

(b) a group of contiguous memory locations ✅

(c) a single memory cell

(d) a pointer variable

32. Which of the following correctly declares a string variable?

(a) string name[16];

(b) char name;

(c) char name[16]; ✅

(d) str name[16];

33. The last character of every string in C is:

(a) newline character

(b) space

(c) EOF

(d) null character '\0' ✅

34. Which library is used for string handling functions in C?

(a) stdio.h

(b) stdlib.h

(c) string.h ✅

(d) conio.h

35. Which function is used to copy one string into another?

(a) copy()

(b) strcopy()

(c) strcpy() ✅

(d) stringcpy()

36. Which function is used to write a string to a text file in C?

(a) puts()

(b) fwrite()

(c) fputs() ✅

(d) fprintf()

37. Which function is used to read a string from a text file in C?

(a) gets()

(b) fread()

(c) fgets() ✅

(d) fscanf()

38. The return type of the fputs() function is:

(a) char*

(b) void

(c) int ✅

(d) FILE*

39. The fputs() function returns ____ if an error occurs.

(a) 0

(b) NULL

(c) EOF ✅

(d) -1

40. Which of the following is NOT written to the file by fputs()?

(a) characters of string

(b) string data

(c) null terminator (‘\0’) ✅

(d) text content

41. The fputs() function automatically appends a newline character.

(a) True

(b) False ✅

(c) Depends on compiler

(d) Depends on file mode

42. The fgets() function stops reading when:

(a) num-1 characters are read

(b) a newline character is found

(c) end of file is reached

(d) all of these ✅

43. The fgets() function returns _____ if successful.

(a) EOF

(b) NULL

(c) the string pointer (str) ✅

(d) an integer value

44. Which function is used to accept a string from the keyboard?

(a) scanf()

(b) fgets()

(c) gets() ✅

(d) getchar()

45. In text files, a separator is used to:

(a) delete data

(b) join records

(c) separate fields of data ✅

(d) end the file

46. Which symbol is used as a separator between name and telephone number in the example?

(a) :

(b) ,

(c) ! ✅

(d) ;

47. Which function works like printf() but writes formatted data to a file?

(a) sprintf()

(b) fprintf() ✅

(c) fputs()

(d) puts()

48. Which function works like scanf() but reads formatted data from a file?

(a) fscanf() ✅

(b) fgets()

(c) fread()

(d) getc()

49. The fprintf() and fscanf() functions differ from printf() and scanf() because they:

(a) work faster

(b) use binary format

(c) work with files instead of console ✅

(d) cannot format data

50. The main advantage of fprintf() and fscanf() is that they:

(a) work only with strings

(b) support binary files only

(c) make it easy to write/read formatted text data to/from files ✅

(d) do not require file pointers

📍 Important Short Questions:

1. What is the difference between temporary and permanent data in C programs?

Answer:

👉 Temporary data exists only while a program is running and is lost after the program ends.

👉 Permanent data is stored on a storage device (like a file) and remains available even after the program ends.

Example:

- **Temporary:** Values entered by the user during program execution.
- **Permanent:** Data saved in contacts.txt file.

2. Define a file in C and explain its purpose.

Answer:

👉 A file is a collection of related records stored on a permanent storage device.

Purpose: Files are used to store data permanently, allowing it to be retrieved and manipulated later.

Example: marks.txt storing students' exam marks.

3. What is a stream in C? Name its two types.

Answer:

👉 A stream is a logical connection between a program and a file, used for reading or writing data.

Types of streams:

1. **Text stream –** sequence of characters.
2. **Binary stream** – sequence of bytes.

4. Explain the difference between a text stream and a binary stream.

Answer:

Feature	**Text Stream**	**Binary Stream**
Data type	Characters	Bytes
Translation	May occur	No

	(e.g., newline to carriage return)	translation
Size	May differ from actual file size	One-to-one with file size

5. What is the role of the EOF marker in a text file?

Answer:

👉 EOF (End of File) marks the end of a text file, signaling functions like getc() to stop reading.

Example: Reading contacts.txt stops when EOF is reached.

6. Which function is used to open a file in C and what are its parameters?

Answer:

👉 The function fopen() opens a file.

Syntax: FILE* fopen(const char* filename, const char* mode);

Parameters:

1. **filename** – Name or path of the file.

2. **mode** – Operation mode (r, w, a, etc.).

7. List all the standard file opening modes in C and briefly explain any two.

Answer:

Modes: r, w, a, r+, w+, a+

Explanation:

- r – Opens an existing file for reading only.
- w – Opens a file for writing; creates if not exist, overwrites if exist.

8. What is a file pointer in C and how is it declared?

Answer:

👉 A file pointer is a variable of type FILE* that points to a file stream.

Declaration:

FILE *fp;

9. How does a pointer differ from a normal variable in C?

Answer:

👉 A normal variable stores a value directly.

👉 A pointer stores the memory address of another variable.

Example:

int num = 25;

int *ptr = # // ptr stores address of num

10. In the given pointer example, what does the *var expression represent?

Answer:

👉 *var gives the value stored at the memory location pointed to by var.

Example:

If var = &num and num = 25, then *var = 25.

11. Which function is used to close a file in C and what is its syntax?

Answer:

👉 The fclose() function is used to close a file.

Syntax:

int fclose(FILE* fp);

Purpose: It disassociates the file from the stream and releases resources.

12. What does the fclose() function return if the file is closed successfully or if an error occurs?

Answer:

👉 Returns 0 if the file is closed successfully.

👉 Returns EOF if an error occurs.

13. Which functions are used to read and write single characters in a file?

Answer:

- getc(FILE* fp) – Reads the next character from a file.
- putc(int ch, FILE* fp) – Writes a character to a file.

14. What value does getc() return when the end of a file is reached?

Answer:

👉 getc() returns EOF when the end of the file is reached or if an error occurs.

15. How are strings stored in C and why is the concept of arrays necessary for string handling?

Answer:

👉 Strings in C are stored as arrays of characters.

👉 Arrays provide contiguous memory locations to store each character of a string individually.

16. How do you declare a string variable in C? Give an example.

Answer:

👉 A string is declared as a char array.

Example:

char name[16]; // Can store a string up to 15 characters + '\0'

17. What is the purpose of the null character \0 in a string?

Answer:

👉 The null character \0 marks the end of the string in C.

Purpose: It tells functions where the string ends, allowing variable-length strings.

18. How can a string be initialized while declaring it? Give an example.

Answer:

👉 Strings can be initialized using double quotes at the time of declaration.

Example:

```
char name[16] = "Lahore";

// 'L', 'a', 'h', 'o', 'r', 'e', '\0'
```

19. Why can’t a string variable be assigned a value directly using the = operator?

Answer:

👉 Strings are arrays of characters, not a single memory location.

👉 Each character must be copied individually, so direct assignment causes an error.

20. Which library function is used to copy one string to another in C? Give its syntax.

Answer:

👉 The strcpy() function is used to copy strings.

Syntax:

char* strcpy(char* dest, const char* source);

Example:

strcpy(name, "I love Pakistan");

21. Which functions are used to write and read strings from a text file in C?

Answer:

👉 fputs() – Writes a string to a file.

👉 fgets() – Reads a string from a file.

22. What value does fputs() return if the operation is successful or fails?

Answer:

👉 Returns EOF if an error occurs.

👉 Returns a non-negative value if successful.

23. How does fgets() determine when to stop reading a string from a file?

Answer:

👉 **fgets() stops reading when:**

1. num-1 characters are read, or
2. A newline character \n is encountered, or
3. The end of file (EOF) is reached.

24. Which function is used to copy user input from the keyboard into a string variable in C?

Answer:

👉 The gets() function is used to read a string from the keyboard and store it in a string variable (char array).

25. What are fprintf() and fscanf() used for in C file handling?

Answer:

👉 fprintf() – Writes formatted data to a file (like printf for files).

👉 fscanf() – Reads formatted data from a file (like scanf for files).

Advantage: They allow easy handling of multiple data types in text files using formatted I/O.

📍 Exercise 14c

1. Fill in the blanks:

(i) A _____ can store text only.

Answer: Text file

Explain:

👉 A text file is a type of file that contains only readable characters. It stores data in the form of characters and can be opened and edited using text editors like Notepad.

(ii) EOF stands for _____.

Answer: End of File

Explain:

👉 EOF is a special marker used in C to indicate the end of a file. It tells the program that there is no more data to read from the file.

(iii) The _____ function is used to open a file.

Answer: fopen()

Explain:

👉 The fopen() function in C is used to open a file and associate it with a stream. It takes two parameters: the file name and the mode in which the file is to be opened (e.g., "r", "w", "a").

(iv) An opened file must be _____ before terminating the program.

Answer: closed

Explain:

👉 Every file opened using fopen() must be closed using fclose() to free the memory allocated to the file pointer and to ensure all data is properly written to the file.

(v) A file opened in _____ mode can be read and appended.

Answer: a+

Explain:

👉 The mode "a+" opens a file for both reading and appending. If the file does not exist, it is created. New data is always added to the end of the file.

(vi) A file pointer is a variable of type _____.

Answer: FILE

Explain:

👉 A file pointer is a variable that stores the address of a file stream. It is declared using FILE* fp; and is used to access files in C.

(vii) A pointer is a memory location whose contents points to _____ memory location.

Answer: another

Explain:

👉 A pointer stores the address of another memory location rather than storing a direct value. For example, int* ptr can store the address of an integer variable.

(viii) n C, every valid string ends with a _____.

Answer: null character (\0)

Explain:

👉 The null character \0 is used to indicate the end of a string in C. This allows functions to know where the string finishes.

(ix) A string is an _____ of characters.

Answer: array

Explain:

👉 In C, a string is stored as an array of characters. Each character occupies one memory location, and the string is terminated with a null character \0.

(x) The fopen() returns a _____ if it fails to open a file for some reason.

Answer: NULL pointer

Explain:

👉 If fopen() cannot open a file (e.g., file does not exist), it returns NULL. This is used to check for errors before performing operations on the file.

2. Choose the correct option:

(i) A file is stored in:

a) kRAM

b) hard disk ✅

c) ROM

d) cache

(ii) Which of the following mode opens only an existing file for both reading and writing:

a) "w"

b) "w+"

c) "r+" ✅

d) "a+"

(iii) Which of the following functions is used to write a string to a file?

a) puts()

b) putc()

c) fputs() ✅

d) fgets()

(iv) On successfully closing a file, the fclose() returns:

a) NULL

b) 0 (zero) ✅

c) 1 (one)

d) FILE pointer

(v) An array subscript should be:

a) int ✅

b) float

c) double

d) an array

3. Write T for true and F for false statement.

(i) A picture can not be stored in a text file. ✅ **T**

(ii) EOF marks the end of a string. ❌ **F**

Correct: EOF marks the end of a file, not a string.

(iii) A null character marks the end of a text file. ❌ **F**

Correct: Null character \0 marks the end of a string, not a file.

(iv) Text files are stored in a FILE* (file pointer). ✅ **T**

(v) The name of the array points to its first element. ✅ **T**

(vi) Array subscript is used to access array elements. ✅ **T**

(vii) An array of characters can store data of any data type. ❌ **F**

Correct: An array of characters can store only characters (strings).

(viii) A binary file is a group of contiguous memory locations. ❌ **F**

Correct: A binary file stores data in binary format on disk, not necessarily contiguous memory.

(ix) C can handle text files only. ❌ **F**

Correct: C can handle both text and binary files.

(x) When an existing file is opened in "w" mode, its contents are over-written. ✅ **T**

🌟 Q.4: Can a file be used for both input and output by the same program?

❖ **Answer:**

Yes, a file can be used for both input (reading) and output (writing) by the same program in C. This is done by opening the file in read and write mode using specific file opening modes.

Explanation:

1. File Opening Modesto for Input and Output:

In C, files can be opened in different modes. To allow both reading and writing on the same file, the following modes are used:

Mode	Description
"r+"	Open an existing file for reading and writing. The file must already exist.
"w+"	Open a file for reading and writing, but it overwrites the existing file. If the file does not exist, a new file is created.
"a+"	Open a file for reading and appending. If the file does not exist, a new file is created.

2. How it Works:

- When a file is opened in "r+" mode, the program can first read the existing data from the file and then write new data to it.
- When a file is opened in "w+" mode, all previous data in the file is erased. The program can write new data and also read it.
- When a file is opened in "a+" mode, new data is appended at the end of the file, and the existing data can still be read.

Example in C:

```
#include <stdio.h>

void main() {
    FILE *fp;
    char ch;
    // Open file for both reading and writing
    fp = fopen("example.txt", "r+");
    if(fp == NULL) {
        printf("File could not be opened.\n");
        return;
    }
    // Read and display existing content
    printf("Existing file content:\n");
    while((ch = getc(fp)) != EOF) {
        putchar(ch);
    }
```

```
    // Move the cursor to the end to write new data
    fseek(fp, 0, SEEK_END);
    fprintf(fp, "\nNew line added.");
    fclose(fp);
    printf("\nData added successfully!\n");
}
```

Explanation of the Program:

- The file is opened in "r+" mode, allowing both reading and writing.
- getc() reads each character until EOF and displays the content on the screen.
- fseek(fp, 0, SEEK_END) moves the file pointer to the end of the file so that new data does not overwrite existing data.
- fprintf() writes new content to the file.

Key Points:

1. A file can be used for both input and output by opening it in a read/write mode.
2. **"r+"** is for reading and writing existing files.

3. **"w+"** is for reading and writing, but it erases existing content.
4. **"a+"** allows reading and appending data at the end of the file.
5. Always close the file using fclose() to save changes and release system resources.

✦ **Summary:**

- Files in C are flexible and can be used for both input and output.
- Correct file mode is essential for the intended operation.
- Helps programs store and update permanent data efficiently.

🌟 **Q.5: What is a stream? Illustrate the difference between text and binary streams.**

❖ **Answer:**

A stream in C is a logical connection between a program and a file or an input/output device. It acts as a data channel that allows a program to read data from a source (like a file or keyboard) or write data to a destination (like a file or screen). Every stream is associated with a file pointer (FILE*) in C.

✦ Explanation:

Types of Streams:

1. Text Stream:

A text stream stores data as a sequence of characters. Some character translations may occur, for example, the newline character \n might be converted to a carriage return/linefeed combination on some systems. Text streams are used for text files such as .txt files. When reading or writing text streams, C interprets the data as characters according to the encoding standard (like ASCII). The end of the file is marked by EOF, and the size of data may vary due to translations.

Example of Text Stream:

```
#include <stdio.h>

int main() {

    FILE *fp = fopen("file.txt", "w");

    fputs("Hello World\n", fp); // Write text

    fclose(fp);

    return 0;
```

```
}
```

2. Binary Stream:

A binary stream stores data as a sequence of bytes. Unlike text streams, no translations occur, so what is written to the file is exactly what is read. Binary streams are used for non-text files like images, audio files, videos, and executables. They are more efficient when working with non-textual data because the storage is exact, byte for byte.

Example of Binary Stream:

```
#include <stdio.h>

int main() {
    FILE *fp = fopen("image.bmp", "wb");
    unsigned char data[3] = {255, 0, 128}; // binary data
    fwrite(data, sizeof(unsigned char), 3, fp);
    fclose(fp);
    return 0;
}
```

Key Points in Words:

- Text streams store characters and may convert some characters automatically (like newline). They are suitable for plain text files.
- Binary streams store exact bytes without any translation. They are suitable for images, videos, or any file where exact data is required.
- Choosing the correct stream depends on the type of data being handled.
- Both text and binary streams use a file pointer to access the file and can be opened, read/written, and closed using standard C functions.

🌟 Q.6: How many modes are there for opening a file in C? Discuss characteristics of different file opening modes.

❖ Answer:

In C, a file can be opened in six different modes, each serving a specific purpose for reading, writing, or appending data. These modes determine how the file is accessed and what operations are allowed.

❖ File Opening Modes in C:

1. "r" – Read Mode:

- Open a file for reading only.
- The file must already exist, otherwise fopen() returns NULL.
- Writing to the file in this mode is not allowed.

2. "w" – Write Mode:

- Open a file for writing only.
- If the file already exists, its contents are overwritten.
- If the file does not exist, it is created.
- Use this mode to create new files or erase old content.

3. "a" – Append Mode:

- Open a file for adding data at the end.
- If the file exists, new data is added after the existing content.
- If the file does not exist, it is created.
- The existing data in the file is never overwritten.

4. "r+" – Read and Write Mode:

- Opens an existing file for both reading and writing.
- The file must exist, otherwise fopen() returns NULL.

- Data can be read and modified anywhere in the file.

5. "w+" – Write and Read Mode:

- Opens a file for both reading and writing.
- If the file exists, its content is overwritten.
- If the file does not exist, it is created.
- Use this mode when you want a fresh file for both reading and writing.

6. "a+" – Append and Read Mode:

- Opens a file for reading and appending.
- If the file exists, new data is added at the end.
- If the file does not exist, it is created.
- Existing data can be read, but writing always happens at the end.

Key Points:

- "r" and "r+" require the file to exist.
- "w" and "w+" overwrite existing files or create new ones.
- "a" and "a+" append data without deleting existing content.
- "+" in a mode indicates both reading and writing are allowed.

🌟 Q.7: What is a file pointer? Briefly explain the concept.

❖ Answer:

A file pointer in C is a special type of pointer that is used to access and manipulate a file. It is a variable of type FILE, which is defined in the standard header file stdio.h. A file pointer acts as a link or reference between the program and the file on the storage device (like a hard disk).

✦ Concept of a File Pointer:

1. Declaration:

A file pointer is declared like this:

FILE *fp;

Here, fp is a pointer variable that will point to the file structure.

2. Purpose:

- It tells the program where the file is located in memory.
- All file operations like reading, writing, appending, or closing a file are performed through the file pointer.

3. Working:

- When you open a file using fopen(), it returns a file pointer.
- If the file cannot be opened, fopen() returns NULL.
- Once you have a valid file pointer, functions like getc(), putc(), fscanf(), fprintf(), etc., can be used to perform operations on that file.

4. Analogy:

- You can think of a file pointer as a remote control. Just as a remote controls a TV without touching it, a file pointer allows your program to control and access a file without directly manipulating its data on disk.

Example of File Pointer Usage:

```
#include <stdio.h>

void main() {

    FILE *fp;

    // Open a file for writing

    fp = fopen("example.txt", "w");

    if (fp == NULL) {
```

```
        printf("Error opening file!\n");
    } else {
        fprintf(fp, "Hello, File Pointer!");
        fclose(fp);  // Close the file
    }
}
```

- **Here**, fp is the file pointer used to write data into example.txt.
- All file operations are performed through fp.

Key Points:

- A file pointer is mandatory for all file operations in C.
- It stores the address of the file structure, not the file data itself.
- Always close a file using fclose(fp) after operations to free resources.

🌟 Q.8: Write a program to merge the contents of two text files into a third file.

❖ Answer:

```
#include <stdio.h>
#include <stdlib.h>
void main() {
    FILE *file1, *file2, *mergedFile;
    char ch;
    // Open the first file for reading
    file1 = fopen("file1.txt", "r");
    if (file1 == NULL) {
        printf("Cannot open file1.txt\n");
        exit(1);
    }
    // Open the second file for reading
    file2 = fopen("file2.txt", "r");
    if (file2 == NULL) {
        printf("Cannot open file2.txt\n");
        fclose(file1);
```

```
    exit(1);
}
// Open the merged file for writing
mergedFile = fopen("merged.txt", "w");
if (mergedFile == NULL) {
    printf("Cannot create merged.txt\n");
    fclose(file1);
    fclose(file2);
    exit(1);
}
// Copy contents of first file to merged file
while ((ch = getc(file1)) != EOF) {
    putc(ch, mergedFile);
}
// Copy contents of second file to merged file
while ((ch = getc(file2)) != EOF) {
```

```
        putc(ch, mergedFile);
    }
    printf("Files merged successfully into merged.txt\n");
    // Close all files
    fclose(file1);
    fclose(file2);
    fclose(mergedFile);
}
```

Explanation:

1. The program opens two source files (file1.txt and file2.txt) in read mode.
2. It opens the third file (merged.txt) in write mode to store the merged content.
3. The getc() function reads one character at a time from the source files.
4. The putc() function writes each character to the merged file.
5. **Two loops are used sequentially:** the first copies file1.txt content, the second copies file2.txt content.

6. fclose() is used to properly close all three files after operations.

🌟 **Q.9: Write a program that counts the total number of characters in a text file. [Note: consider the blank space a character]**

❖ **Answer:**

```
#include <stdio.h>

#include <stdlib.h>

void main() {

    FILE *file;

    char ch;

    int count = 0;

    // Open the file in read mode

    file = fopen("input.txt", "r");

    if (file == NULL) {

        printf("Cannot open input.txt\n");

        exit(1);
```

```
    }

    // Read each character until end of file

    while ((ch = getc(file)) != EOF) {

        count++;  // Increment count for every character including
spaces

    }

    printf("Total number of characters in the file: %d\n", count);

    // Close the file

    fclose(file);

}
```

Explanation:

1. The program opens a file named input.txt in read mode using fopen().
2. A variable count is initialized to zero to keep track of characters.
3. The getc() function reads one character at a time from the file.

4. Every time a character is read, count is incremented by 1, including spaces, tabs, and punctuation.
5. When getc() reaches the EOF (end of file), the loop stops.
6. The total number of characters is printed using printf().
7. **Finally**, the file is closed using fclose() to free system resources.

🌟 **Q.10: Write a program that counts the number of words in a text file and displays the count on the screen.**

❖ **Answer:**

```
#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

void main() {

   FILE *file;

   char ch;

   int words = 0;

   int inWord = 0;  // Flag to check if we are inside a word

   // Open the file in read mode
```

```
file = fopen("input.txt", "r");
if (file == NULL) {
    printf("Cannot open input.txt\n");
    exit(1);
}
// Read each character until end of file
while ((ch = getc(file)) != EOF) {
    if (isspace(ch)) {
        inWord = 0; // Space or newline indicates end of a word
    } else if (inWord == 0) {
        inWord = 1; // Start of a new word
        words++;    // Increment word count
    }
}
printf("Total number of words in the file: %d\n", words);
// Close the file
```

```
    fclose(file);
}
```

Explanation:

1. The program opens a file named input.txt in read mode using fopen().
2. The variable words is initialized to zero to keep track of word count.
3. The variable inWord acts as a flag to detect whether the program is currently reading inside a word.
4. The getc() function reads one character at a time from the file.
5. isspace(ch) checks for spaces, tabs, or newline characters. When a space is found, it means the current word ended, and inWord is set to 0.
6. If a non-space character is found and inWord is 0, this indicates the start of a new word, so the word count is incremented and inWord is set to 1.
7. The loop continues until EOF is reached.
8. **Finally**, the program prints the total number of words and closes the file using fclose().

Note:

This chapter is designed to provide a solid foundation of knowledge, with the goal of deepening understanding and encouraging further exploration of the subject. The content has been carefully selected to support effective learning and inspire students to engage with the topic more deeply.

Author: Muhammad Asghar

Purpose: To contribute to education by offering insightful, valuable content that enhances learning and understanding.