**FORMAN CHRISTIAN COLLEGE (A CHARTERED UNIVERSITY)**

****

**COMP 451 (Compiler Construction)**

**2022 FALL**

**Lab 1 - A**

**Muhammad Hamza (231-450349)**

**Gulraiz Noor Bari (231-525536)**

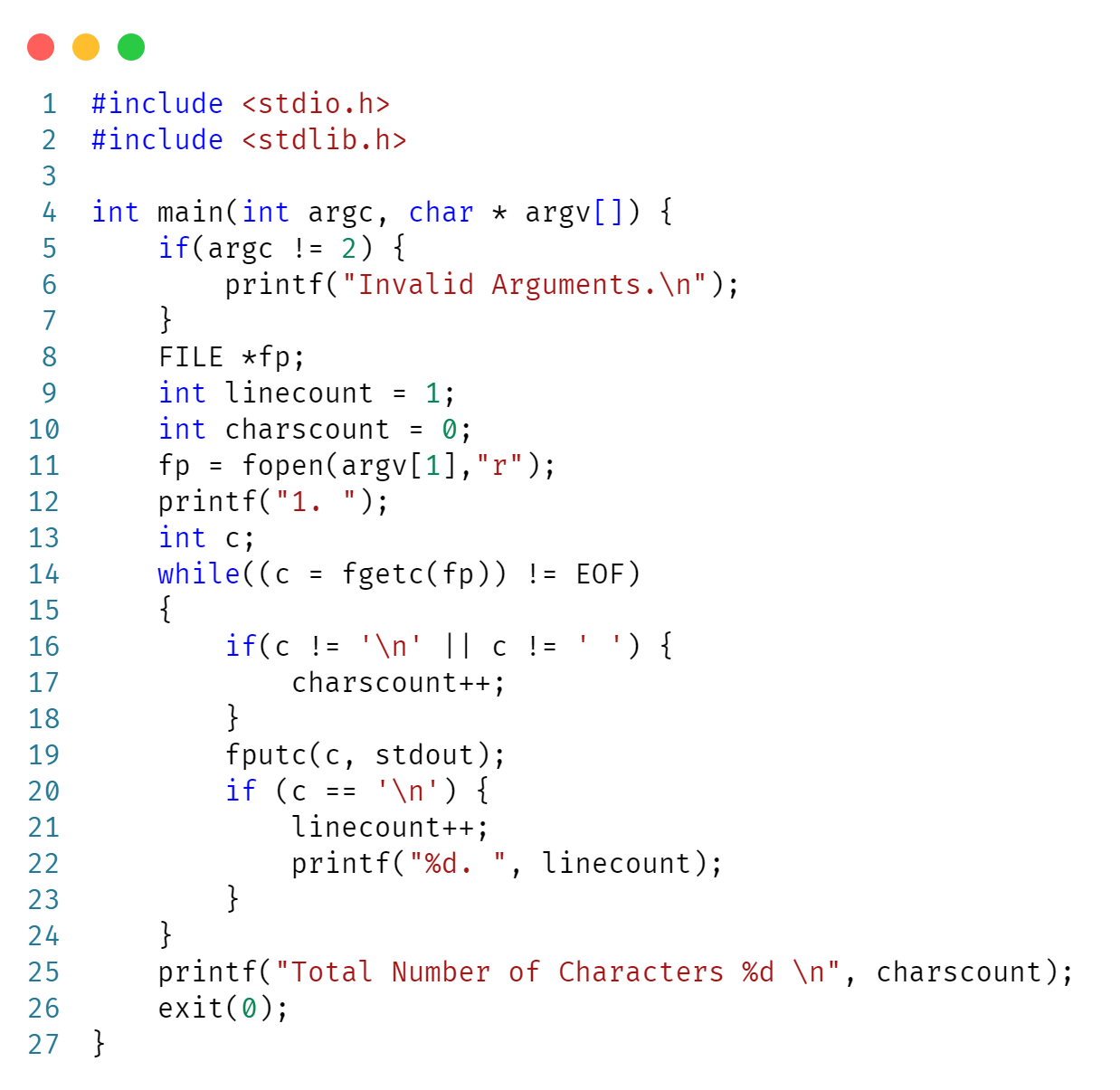
**INTRODUCTION:**

* **stdio.h** (standard input/output)is a header file that contains declarations for functions like, printf, scanf, etc.
* **stdlib.h** (standard library) is a header file that contains declarations of functions that involves memory allocation and process control. For example, in our program we used exit(0) at the end; which means to successfully terminate the program.
* **main():** The main function serves as the starting point of the program execution in C language. User can pass any number of parameters depending upon the requirements of the program logic or structure. We have passed two parameters:
* **int argc:** Keeps count of the number of command-line arguments entered by the user that also includes the name of the program.
* **char \* argv[]:** Here argv (stores command-line arguments) is an array of pointers that is pointing to the characters of the command-line arguments entered by the user.

**LOGIC/ALGORITHM:**

The code is a program that reads a file specified by the user, character by character and displays it on the command-line, with each line preceding by a line number and displays number of characters in the file.

* The program starts by checking whether two arguments were passed by the user in command line, which is done by argc (which has the count of arguments entered in the command-line). If the number of arguments is not two the function is not proceeded and displays a message “Invalid argument”, otherwise if user has entered two arguments; in our case the executable and file name that is to be read, the program proceeds.
* Two counters are initialized:
* **linecount:** Keeps count of the number of lines in the text file.
* **charscount:** Keeps count of the number of characters in the text file.
* Then a file is opened using **fopen** function which takes two parameters; filename and the mode in which the file should be opened, it returns a pointer pointing to the address in the memory where the file is stored. In our case “fp” holds that file pointer that points to the file stored in argv[1], as the file to read is the second argument user enters in the command-line.
* The file is read by each character in a loop using **fgetc** function, which returns the ASCII code of each character and is stored in a integer type variable “c”. The loop will run until End of File.
* In every iteration of the loop/for every character in the file we perform the following checks/operations:
* We check whether the character stored in “c” equals to “**\n**” (newline) or a space (“ ”), if not then **charcount** is incremented by 1.
* After reading one character we use fputc(c, stdout) which prints each character on the output; which is the command-line.
* If at any point of the program character being read becomes equal to the “**\n**”, the line counter is incremented by 1 and **linecount** is displayed on the command-line preceding each line (from the file being read).
* The loop runs until the character read (“c”) becomes equal to EOF (End of File) and the contents of the file are read by each character and displayed on the command-line preceding by the line number.
* After the loop ends, a line “Total Number of characters in the file” is printed and followed by “%d” (decimal formatter), which replaces **charcount** (stores number of characters in the file).
* The program ends with a exit(0) statement, which refers to successful termination of the program.

**CODE:**

**Text

Description automatically generatedOutput**

**Text

Description automatically generatedText

Description automatically generated**