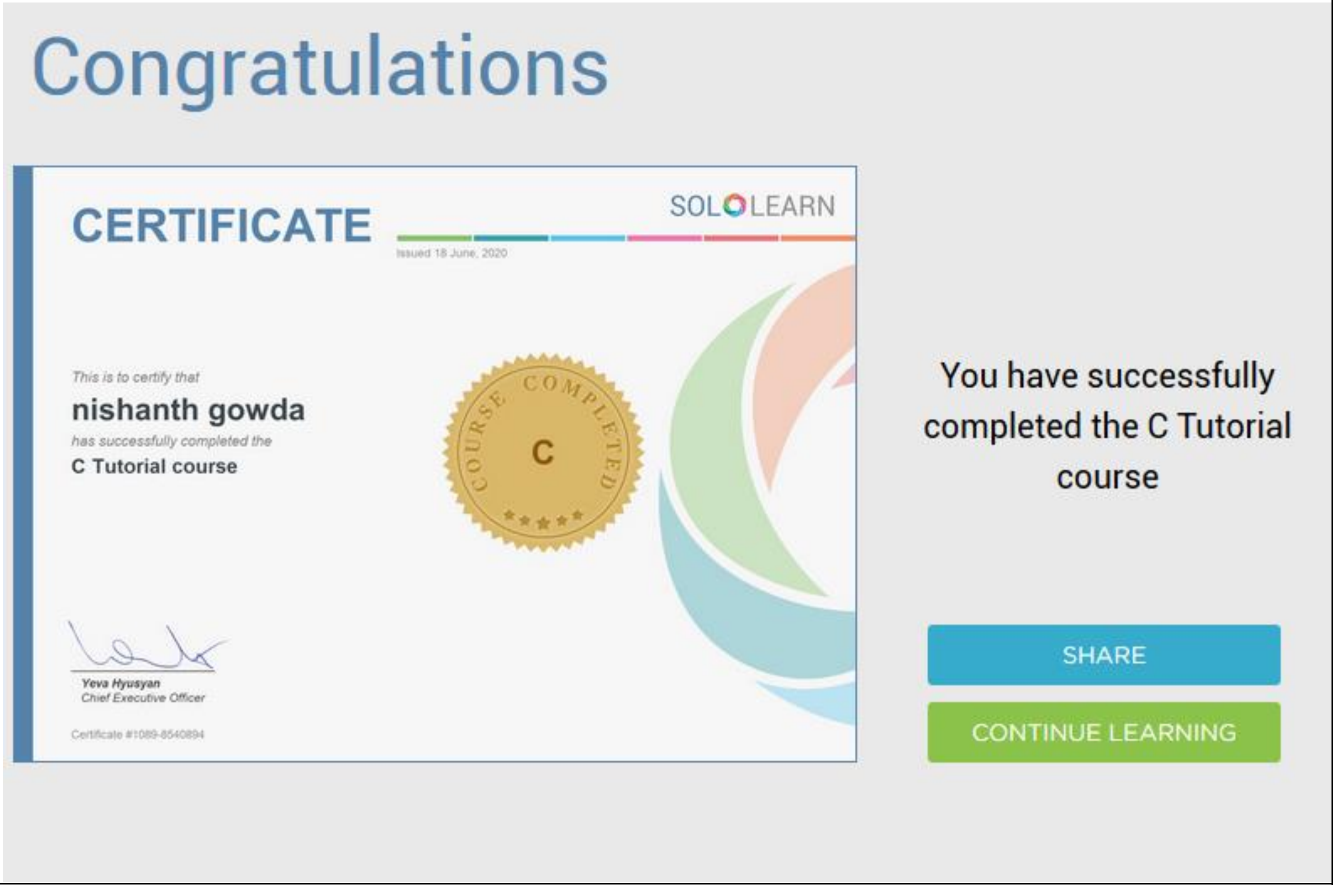
# DAILY ASSESSMENT FORMAT

| | | | |
|---|---|---|---|
| **Date:** | **18/June/2020** | **Name:** | **nishanth** |
| **Course:** | **C programming** | **USN:** | **4al17ec063** |
| **Topic:** | 1.Files & Error Handling<br>**2.** The Processors | **Semester & Section:** | **6th b** |
| **GitHub Repository:** | **nishanthvr** | | |

**FORENOON SESSION DETAILS**

**Image of session:**



Congratulations

CERTIFICATE

SOLOLEARN

Issued 18 June, 2020

*This is to certify that*

**nishanth gowda**

*has successfully completed the*

**C Tutorial course**

COURSE COMPLETED

C

*Yeva Hyusyan*
*Chief Executive Officer*

Certificate #1089-8540894

You have successfully completed the C Tutorial course

# Accessing Files

An external file can be opened, read from, and written to in a C program. For these operations, C includes the **FILE** type for defining a file stream. The **file stream** keeps track of where reading and writing last occurred.

The **stdio.h** library includes file handling functions:
**FILE** Typedef for defining a file pointer.

**fopen(filename, mode)** Returns a FILE pointer to file *filename* which is opened using *mode*. If a file cannot be opened, NULL is returned.

Mode options are:
- **r** open for reading (file must exist)
- **w** open for writing (file need not exist)
- **a** open for append (file need not exist)
- **r+** open for reading and writing from beginning
- **w+** open for reading and writing, overwriting file
- **a+** open for reading and writing, appending to file

**fclose(fp)** Closes file opened with FILE fp, returning 0 if close was successful. **EOF** (end of file) is returned if there is an error in closing.

The following program opens a file for writing and then closes it:

```
#include <stdio.h>

int main() {
FILE *fptr;

fptr = fopen("myfile.txt", "w");
if (fptr == NULL) {
printf("Error opening file.");
return -1;
}
fclose(fptr);
return 0;
}
```

## Reading from a File

The **stdio.h** library also includes functions for reading from an open file. A file can be read one character at a time or an entire string can be read into a character **buffer**, which is typically a char array used for temporary storage.

**fgetc(fp)** Returns the next character from the file pointed to by *fp*. If the end of the file has been reached, then **EOF** is returned.

**fgets(buff, n, fp)** Reads n-1 characters from the file pointed to by *fp* and stores the string in buff. A NULL character '\0' is appended as the last character in *buff*. If fgets encounters a newline character or the end of file before n-1 characters is reached, then only the characters up to that point are stored in buff.

**fscanf(fp, conversion_specifiers, vars)** Reads characters from the file pointed to by *fp* and assigns input to a list of variable pointers *vars* using *conversion_specifiers*. As with scanf, fscanf stops reading a string when a space or newline is encountered.

The following program demonstrates reading from a file:

```
#include <stdio.h>

int main() {
```

```
FILE *fptr;
int c, stock;
char buffer[200], item[10];
float price;

/* myfile.txt: Inventory\n100 Widget 0.29\nEnd of List */

fptr = fopen("myfile.txt", "r");

fgets(buffer, 20, fptr); /* read a line */
printf("%s\n", buffer);

fscanf(fptr, "%d%s%f", &stock, item, &price); /* read data */
printf("%d %s %4.2f\n", stock, item, price);

while ((c = getc(fptr)) != EOF) /* read the rest of the file */
printf("%c", c);

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

## Binary File I/O

Writing only characters and strings to a file can become tedious when you have an array or structure. To write entire blocks of memory to a file, there are the following binary functions:

Binary file mode options for the fopen() function are:
- **rb** open for reading (file must exist)
- **wb** open for writing (file need not exist)
- **ab** open for append (file need not exist)
- **rb+** open for reading and writing from beginning
- **wb+** open for reading and writing, overwriting file
- **ab+** open for reading and writing, appending to file

**fwrite(ptr, item_size, num_items, fp)** Writes *num_items* items of *item_size* size from pointer *ptr* to the file pointed to by file pointer *fp*.

**fread(ptr, item_size, num_items, fp)** Reads *num_items* items of *item_size* size from the file pointed to by file pointer *fp* into memory pointed to by *ptr*.

**fclose(fp)** Closes file opened with file *fp*, returning 0 if close was successful. **EOF** is returned if there is an error in closing.
**feof(fp)** Returns 0 when the end of the file stream has been reached.