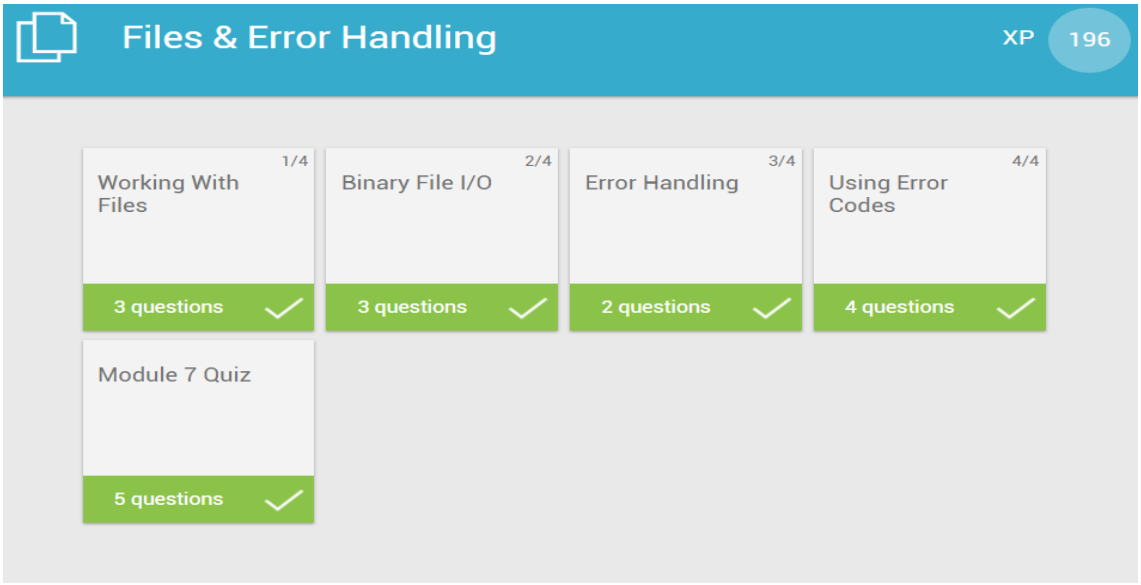
DAILY ASSESSMENT FORMAT

Date:	20-06-2020	Name:	POOJA K S
Course:	C programming	USN:	4AL17EC070
Topic:	Files & Error Handling	Semester & Section:	6th sem & 'B' sec
Github Repository:	pooja-shivanna		

FORENOON SESSION DETAILS
Image of session



Report –

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.

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

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.

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;
}
```

Writing to a File

The stdio.h library also includes functions for writing to a file. When writing to a file, newline characters '\n' must be explicitly added.

fputc(char, fp) Writes character char to the file pointed to by fp.

fputs(str, fp) Writes string str to the file pointed to by fp.

fprintf(fp, str, vars) Prints string str to the file pointed to by fp. str can optionally include format specifiers and a list of variables vars.

The following program demonstrates writing to a file:

```
FILE *fptr;
char filename[50];
printf("Enter the filename of the file to create: ");
gets(filename);
```

```
fptr = fopen(filename, "w");


/* write to file */

fprintf(fptr, "Inventory\n");

fprintf(fptr, "%d %s %f\n", 100, "Widget", 0.29);

fputs("End of List", fptr);
```

Exception Handling

An exception is any situation that causes your program to stop normal execution. Exception handling, also called error handling, is an approach to processing runtime errors.

C does not explicitly support exception handling, but there are ways to manage errors:

- Write code to prevent the errors in the first place. You can't control user input, but you can check to be sure that the user entered valid input. When performing division, take the extra step to ensure that division by 0 won't occur.

The exit Command

The exit command immediately stops the execution of a program and sends an exit code back to the calling process. For example, if a program is called by another program, then the calling program may need to know the exit status.

The feof and ferror Functions

In addition to checking for a NULL file pointer and using errno, the feof() and ferror() functions can be used for determining file I/O errors:

feof(fp) Returns a nonzero value if the end of stream has been reached, 0 otherwise. feof also sets EOF.

ferror(fp) Returns a nonzero value if there is an error, 0 for no error.

```
FILE *fptr;
```

```
int c;


errno = 0;
fptr = fopen("myfile.txt", "r");
if (fptr == NULL) {
fprintf(stderr, "Error opening file. %s\n", strerror(errno));
exit(EXIT_FAILURE);
}
while ((c = getc(fptr)) != EOF) /* read the rest of the file */
printf("%c", c);

if (ferror(fptr)) {
printf("I/O error reading file.");
exit(EXIT_FAILURE);
}
else if (feof(fptr)) {
printf("End of file reached.");
}
```

CERTIFICATE

SOLOLEARN

Issued 19 June, 2020

This is to certify that

POOJA K S

has successfully completed the

C Tutorial course

Yeva Hyusyan
Chief Executive Officer

Certificate #1089-18849166