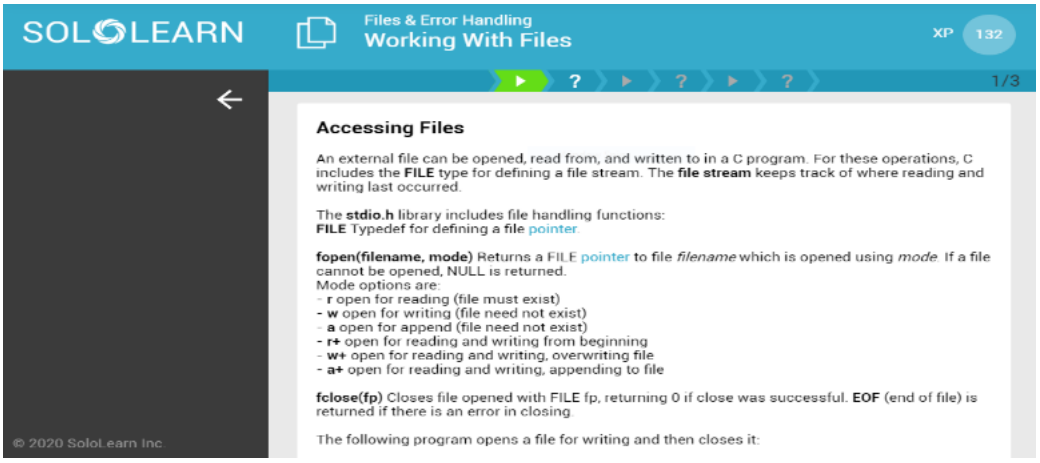
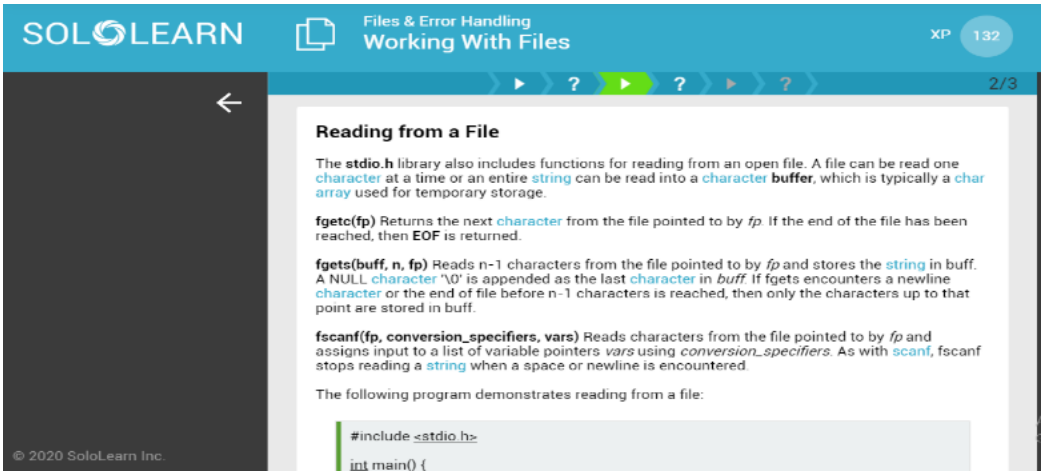
DAILY ASSESSMENT FORMAT

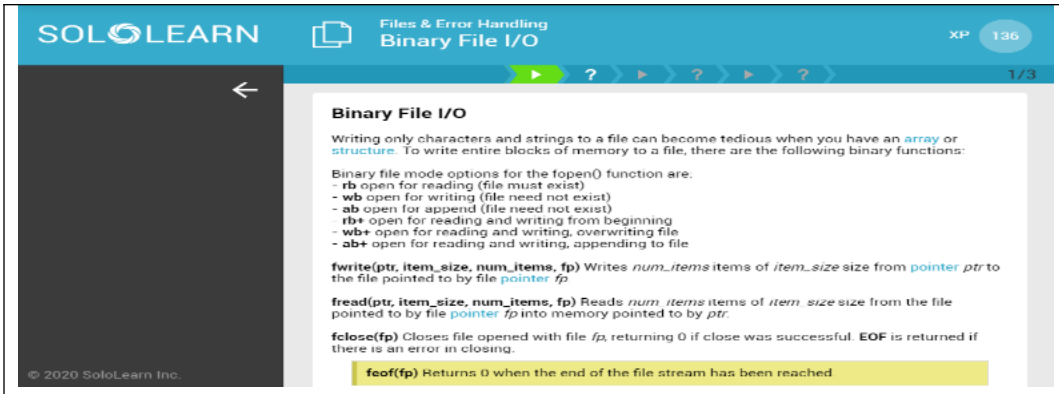
Date:	20-06-2020	Name:	Rajeshwari Gadagi
Course:	Solo Learn	USN:	4AL17EC076
Topic:	C Programming	Semester & Section:	6th SEM & 'B' SEC
Github Repository:	Rajeshwari-gadagi		

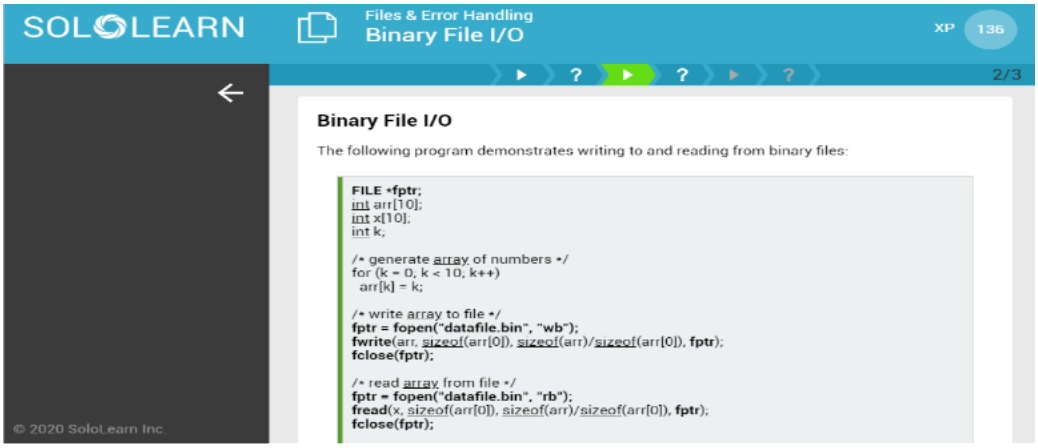
FORENOON SESSION DETAILS

Image of session






SOLOLEARN
Files & Error Handling
Binary File I/O
XP 136
1/3
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
© 2020 SoloLearn Inc.


SOLOLEARN
Files & Error Handling
Binary File I/O
XP 136
2/3
Binary File I/O
The following program demonstrates writing to and reading from binary files:
FILE *fptr;
int arr[10];
int x[10];
int k;
/* generate array of numbers */
for (k = 0; k < 10; k++)
arr[k] = k;
/* write array to file */
fptr = fopen("datafile.bin", "wb");
fwrite(arr, sizeof(arr[0]), sizeof(arr)/sizeof(arr[0]), fptr);
fclose(fptr);
/* read array from file */
fptr = fopen("datafile.bin", "rb");
fread(x, sizeof(arr[0]), sizeof(arr)/sizeof(arr[0]), fptr);
fclose(fptr);
© 2020 SoloLearn Inc.

C Programming

Aug 20/08/2020 - Saturday

→ File Handling

Working with Files

> Accessing Files

An external file can be opened, read from, written to in a C program.

The stdio.h library provides file handling functions

FILE typedef for declaring a file pointer

fopen (filename, mode)

fclose(fp).

Mode options are:

- r open for reading (file must exist)
- w open for writing (file need not exist)
- a open for appending (file need not exist)
- r+ open for reading & writing from beginning
- w+ open for reading & writing, overwriting file
- a+ open for reading & writing, appending to file

> Reading from a file

The stdio.h library includes functions for reading an open file

fgetc(fp)

fgets(buffer, n, fp)

fscanf(fp, conversion specifiers, vars)

> Writing to a file

The stdio.h library also includes functions for writing to a file.

fputc(char, fp)

fputs(str, fp)

fprintf(fp, str, vars)

→ Binary files

Writing only characters and strings to open a file can become difficult, you have to store other binary file modes options to the fopen() function are

- rb open for reading
- wb open for writing
- ab open for append
- rb+ open for reading & writing from beginning
- wb+ open for reading & writing, overwriting file
- ab+ open for reading & writing, appending to file

fwrite(ptr, item_size, num_items, fp)

fread(ptr, item_size, num_items, fp)

fclose(fp).

• Binary file I/O

• Controlling the file pointer

There are functions in stdio.h to controlling the location of the file pointer in a binary file:

ftell(fp)

fseek(fptr, num_bytes, from_pos)

- SEEK_SET - Set pointer to file
- SEEK_CUR - current location
- SEEK_END - end of file

• Error handling

• 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.

• The exit command

The exit command immediately stops the execution of a program & sends an exit code back to the calling process.

> Using Error Codes

• Using errno

Some library functions such as fopen(), return error code, when they do not execute as expected. errno - global name errno.h header file

• The perror and strerror functions

For more descriptive message about the error, you can use perror(). You can also obtain the message using strerror() in the string.h header file.

• EDOM & ERANGE Error Codes

Set errno to the defined macro values EDOM when domain is out of range.

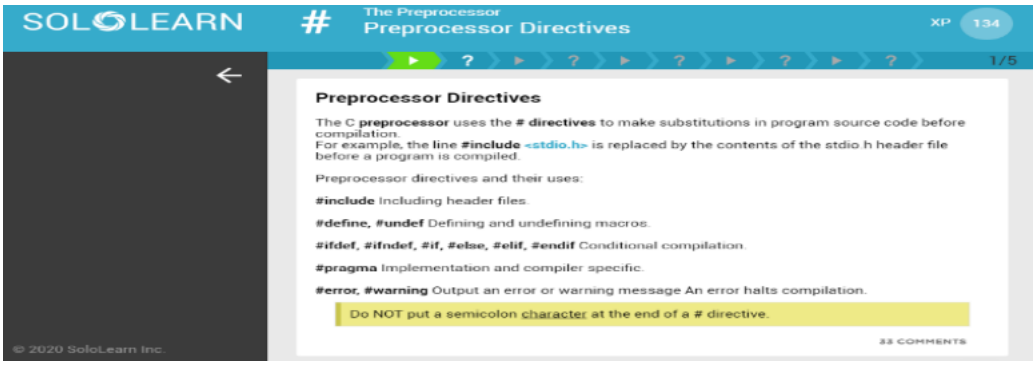
Similarly, the ERANGE macro value is used when there is a range error.

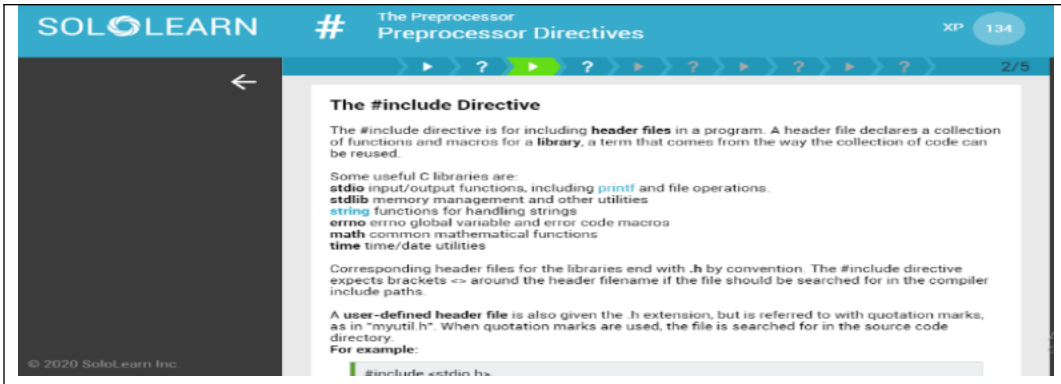
• The feof and ferror functions

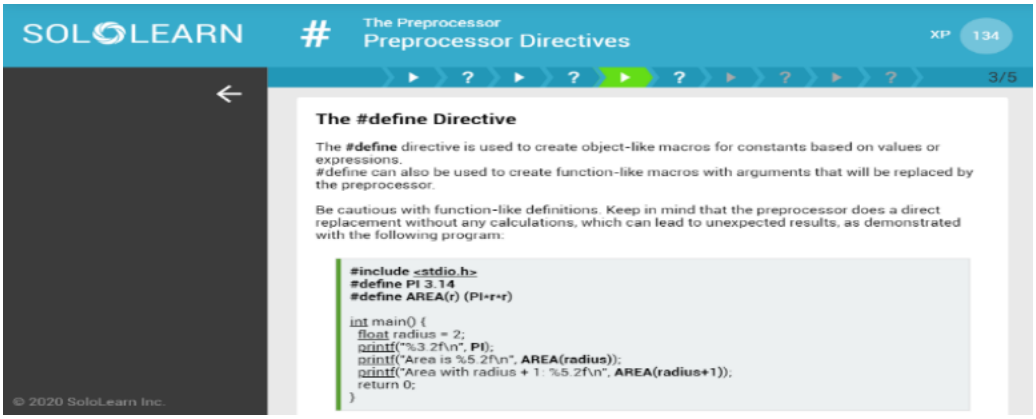
Date:	20-06-2020	Name:	Rajeshwari Gadagi
Course:	Solo Learn	USN:	4AL17EC076
Topic:	C Programming	Semester & Section:	6th SEM & 'B' SEC
Github Repository:	Rajeshwari-gadagi		

AFTERNOON SESSION DETAILS

Image of session




SOLOLEARN
#
The Preprocessor
Preprocessor Directives
XP 134
2/5
The #include Directive
The #include directive is for including header files in a program. A header file declares a collection of functions and macros for a library, a term that comes from the way the collection of code can be reused.
Some useful C libraries are:
stdio input/output functions, including printf and file operations
stdlib memory management and other utilities
string functions for handling strings
errno errno global variable and error code macros
math common mathematical functions
time time/date utilities
Corresponding header files for the libraries end with .h by convention. The #include directive expects brackets <> around the header filename if the file should be searched for in the compiler include paths.
A user-defined header file is also given the .h extension, but is referred to with quotation marks, as in "myutil.h". When quotation marks are used, the file is searched for in the source code directory.
For example:
#include <stdio.h>
© 2020 SoloLearn Inc


SOLOLEARN
#
The Preprocessor
Preprocessor Directives
XP 134
3/5
The #define Directive
The #define directive is used to create object-like macros for constants based on values or expressions.
#define can also be used to create function-like macros with arguments that will be replaced by the preprocessor.
Be cautious with function-like definitions. Keep in mind that the preprocessor does a direct replacement without any calculations, which can lead to unexpected results, as demonstrated with the following program:
#include <stdio.h>
#define PI 3.14
#define AREA(r) (PI*r*r)
int main() {
float radius = 2;
printf("%3.2f\n", PI);
printf("Area is %5.2f\n", AREA(radius));
printf("Area with radius + 1: %5.2f\n", AREA(radius+1));
return 0;
}
© 2020 SoloLearn Inc