CSC111

Administrivia

- No labs next week (week of March 14th)
  - Office hours instead (in all lab sections)
  - Labs resume week of March 21st
- Midterm 2 – March 21st in class
  - Covers material up until and including today's class
  - 70 minutes
  - You are allowed one piece of letter paper with anything on it.
  - New material both days next week (not tested)

Visualizing 2D arrays…

```
int my_ints[2][3] = {{1, 2, 3},
                     {4, 5, 6}};
```

Number of rows

Number of columns

2D array **column** index

my_ints

	0	1	2
0	1	2	3
1	4	5	6

2D array **row** index

A value in the 2D array
At row 1, column 2

```c
#include <stdio.h>

void print_table_ptrs(int num_rows, int num_cols, int table[num_rows][num_cols]);
//main omitted intentionally

/* Purpose: print values in table with dimensions num_rows by num_cols
 * Parameters:   int num_rows, number of rows in table, >=0
 *               int num_cols, number of columns in table, >=0
 *               int table[num_rows][num_cols]
 */
void print_table_ptrs(int num_rows, int num_cols, int table[num_rows][num_cols]) {
        int row, col;

        for(row=0; row<num_rows; row++) {
                for(col=0; col<num_cols; col++) {
                        printf("%d ", table[row][col]);
                }
                printf("\n");
        }
}
```

File I/O
(file input/output)

Directories

- Also known as Folders
- Can access folders on the command prompt
  - Use the cd command to change directories
    - ex cd assignments/
  - Use the ls command to list folders and files
- On Linux(our coding platform)/Mac the ‘/’ character is used to separate folders
- On windows the ’\’ character is used (generally)

- Demo

Steps to access a file for I/O

- Open the file given its name and location
- check to see the file was opened successfully, if it was…
  - Access the file + do operations
  - Close the file
- Otherwise, alert that there is an error

Opening a file with `fopen`

Documentation

```
Purpose:  opens the given filename in the given mode

Parameters: filename, valid path/filename of a textfile
            mode, the mode the file is opened in, ie:
               “r” for reading, “w” for writing, “a” for appending

Returns:   FILE*   a pointer to the file,
           or NULL if the FILE is not opened successfully
```

Opening a file with `fopen`

Examples:

```
FILE* file_handle;

file_handle = fopen(“myfile.txt”, “r”);

file_handle = fopen (“myfile.txt”, “w”);

file_handle = fopen (“myfile.txt”, “a”);

file_handle = fopen(“C:\files\myfile.txt”, “r”);
```

check to see the file was opened successfully

```
FILE* file_handle;
file_handle = fopen("myfile.txt", "r");

if (file_handle == NULL) {
     printf("error opening file\n");
} else {
     // code to read from the file...
}
```

reading from a file with: `fscanf`

Documentation

```
Purpose:  scans input from file pointed to by file_handle
   and stores it in the specified format to the specified address

Parameters: FILE* file_handle – an valid opened file
            a string - containing format specifiers
            pointer types - addresses to store scanned values

Returns: int – the number of values scanned successfully
```

Example

```
int i;
double n;
int scanned;
FILE* file_handle;
// omitted code to open file

scanned = fscanf(file_handle, “%d %lf”, &i, &n);
// scanned is 2 if the file contains 2 valid numbers
```

closing a file with `fclose`

Documentation

```
Purpose:    closes the file pointed to by file_handle
Parameters: FILE* file_handle, a pointer to an opened file
Returns:    nothing
```

Example

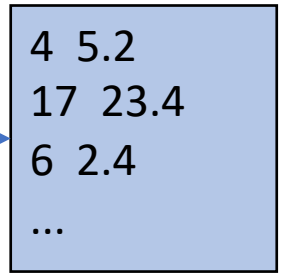
```
FILE* file_handle;
// omitted code to open and read from/write to file
fclose(file_handle);   //closes the file
```

Example reading from a file

```
FILE* file_handle;
int i;
double n;
file_handle = fopen("myfile.txt", "r");

if (file_handle == NULL) {
      printf("error opening file\n");
} else {
      // read into i, n while 2 numbers are successfully read
      while( fscanf(file_handle, "%d %lf", &i, &n)      ) {
            printf("%d, %.2f\n", i, n);
      }
      fclose(file_handle);
}
```

file_handle



OUTPUT:

```
4, 5.20
17, 23.40
6, 2.40
...
```

Example writing to a file

```
FILE* file_handle;
int i;
double limit = 100;
file_handle = fopen("myfile.txt", "w");

if (file_handle == NULL) {
      printf("error opening file\n");
} else {
      // overwrite myfile.txt with numbers 0 to limit-1
      for(i=0; i<limit; i++) {
            fprintf(file_handle, "%d\n", i);
      }
      fclose(file_handle);
}
```

Demo – read and write

Complete the following code:

```
#include <stdio.h>
#define INPUTFILE "input.dat"

int count_above();

int main( void ) {
    printf("Found %d values above the threshold entered\n", count_above());
    return 0;
}

/*
 * Purpose: prompts user for a threshold value and counts the number of entries
 *    in INPUTFILE that are greater than the given threshold value
 * Parameters: None
 * Returns: int, the count, -1 if error opening INPUTFILE or reading from user
 */

int count_above() {
    FILE* in_file;
    double next_val;
    double threshold_val;
    int count_above = 0;

    in_file = fopen( INPUTFILE, "r" );
    if( in_file == NULL ) {
        printf( "Error opening input file\n" );
        return -1;
    } else {        //your code here

    }
}
```

Complete the following function

- Write a function **write_sine_table** that takes a positive integer N. The function should write to the file a table of sine values for each of the following values:

$$\pi/N,\ 2\pi/N,\ 3\pi/N,\ \ldots,\ \pi$$

- For example, if the function is called with an N of 5, it should write the following table of values to the file sineTable.dat
- Remember the function sin(x) in math.h

x	sin(x)
0.628	0.5878
1.257	0.9511
1.885	0.9511
2.513	0.5878
3.142	0.0000

```
#include <stdio.h>
#include <math.h>

#define OUTPUT_FILE "sine_table.dat"
#define PI acos(-1.0)

void write_sine_table(int n);

int main( void ) {
    write_sine_table(5);

    return 0;
}
```