16.216: ECE Application Programming

Summer 2015

Lecture 8: Key Questions
June 12, 2014

1. Explain the use of the `fopen()` function.

2. Explain the use of the `fclose()` function.

3. Explain how `fscanf()` and `fprintf()` are used for formatted file I/O.

4. **Example:** Write a program to:
   - Read three integers from file `myinput.txt`
   - Determine the sum and average of those values
   - Write the original values, sum, and average to file `myoutput.txt`.

5. Explain how `fread()` and `fwrite()` are used for unformatted I/O.

6. Explain how to check that the end of a file has been reached, or if an error has occurred.

7. Describe the functions used for character I/O.

8. Describe the functions used for line I/O.

9. Describe the standard I/O streams and explain how the file I/O functions can be used to write to these locations.

10. **Example:** Show the output of each of the following short program.

a. Input: **`Test Input 1 23 4 5`**

```
void main() {
   char c;
   char buffer[50];
   int i, n;
   i = 0;
   while ((c = fgetc(stdin)) != '\n') {
      if (c != ' ') {
           buffer[i++] = c;
      }
   }
   buffer[i] = '\0';
   fputs(buffer, stdout);
}
```

b. Input:

```
Test1
Test 2
abcdefghijklmnopqrstuvwxyz
This is a test of the fgets() function
```

```
void main() {
   char str[25];
   int i;
   for (i = 0; i < 5; i++) {
      fgets(str, 24, stdin);
      strcat(str, "\n");
      fputs(str, stdout);
   }
}
```

c. Input:

```
1024Some other stuff
```

```
void main() {
   char c;
   char buffer[50];
   int n = 0;

   // isdigit in <ctype.h>
   while (isdigit(c = getchar())) {
      n = n * 10 + (c - 48);   // Hint: '0' = 48   }
      // (ASCII value)
   ungetc(c, stdin);
   fgets(buffer, 50, stdin);

   printf("n = %d, n * 2 = %d\n", n, n * 2);
   printf("buffer = %s\n", buffer);
}
```

For today's programming exercise, you will complete the program below according to the comments listed in the code:

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

FILE *openFile(char *mode);      // Used for opening files;
                                 // exiting prog. if appropriate

void main() {
     int arr[20];     // Input array for use with binary file
     int test;        // Input/output value for formatted I/O

     FILE *fpArr;     // Pointer to file holding array values
     FILE *fpIn;      // Pointer to file with test input values
     FILE *oFile;     // Pointer to output file
     int i;

     // CALL openFile() TO OPEN FILE WITH ARRAY

     // READ CONTENTS OF ARRAY FROM FILE

     // CALL openFile() TO OPEN FILE WITH TEST INPUT VALUES

     // CALL openFile() TO OPEN OUTPUT FILE

     // READ 20 VALUES FROM TEST INPUT FILE
     // FOR EACH ONE, PRINT THE FOLLOWING TO OUTPUT FILE:
     //    <test> + <appropriate array value> = <sum>
     // FOR EXAMPLE, IF FIRST TEST VALUE IS
     //   5 AND ARR[0] = 6, PRINT
     //    5 + 6 = 11

     // CLOSE ANY OPEN FILES
}

FILE *openFile(char *mode) {
     /* COMPLETE THIS FUNCTION SO THAT IT:
         - READS THE NAME OF THE FILE TO BE OPENED
         - OPENS FILE USING THE MODE SPECIFIED AS AN ARGUMENT
         - EXITS PROGRAM IF FILE DOESN'T OPEN
         - RETURNS POINTER TO FILE IF IT DOES OPEN */
}
```

Use this space to complete the `openFile()` function:

```
FILE *openFile(char *mode) {









}
```

Use this space to start the main program: show how you would call openFile() to open the appropriate files, and show how you would read the contents of the array.

Use this space to show how you would complete the program—repeatedly read an input value from the test input file, add that value to the appropriate element from the array, and print the appropriate information to the output file.

After those operations are done, close all open files.