ENGR 0012 – Engineering Problem Solving

Goals for this week:

- Read from a file
  - Print to a file

Please submit your HW!
(Old and new submission systems)

Break and continue are used similarly to how they are used in MATLAB:

- Use break "to exit a loop regardless of the outcome of the relational expression"[1]
- Use continue "to skip one iteration of a loop"[1]
- See your textbook for examples!

1. From course textbook

Instead of scanf and printf, you can use the cin and cout commands

Instead of scanf

Instead of printf

- Must include the iostream library as follows:

```
#include <iostream>    (without the ".h")
using namespace std;
```

- (See examples in your textbook)

Note that we will be focusing on scanf and printf, not cin and cout!

We can open and close files in order to read or write information, as follows:

A pointer is a variable "that is assigned the value of the address of the first memory location for the data file"[1]

- Use the FILE command to define the file pointer:

```
FILE *pointer_variable_ name;
```

Need * to indicate that it is a pointer

The pointer will be assigned an address for the starting location of the file (something like: 12345678), and you can print it with the %p format

- Use the fopen command:

```
pointer_variable_name = fopen("filename", "option");
```

w to write to the file
r to read from a file
a to append to an existing file

From slides by N. Vidic
1. From course textbook

We can open and close files in order to read or write information, as follows:

- When done, close the file with fclose:

```
fclose(outfile);
```

- The file doesn't already have to exist to open it for writing (the program can create a new file)

We can open and close files in order to read or write information, as follows:

- Use fprintf to write data to a file that you have opened:

fprintf(pointer_variable_name,”text and/or data types”,variable names);

Notice this is the same syntax as printf, but we add the pointer variable in the beginning

- Note: w will automatically overwrite any previous data in the file
- To add data to a file without overwriting previous data, use the a option in fopen

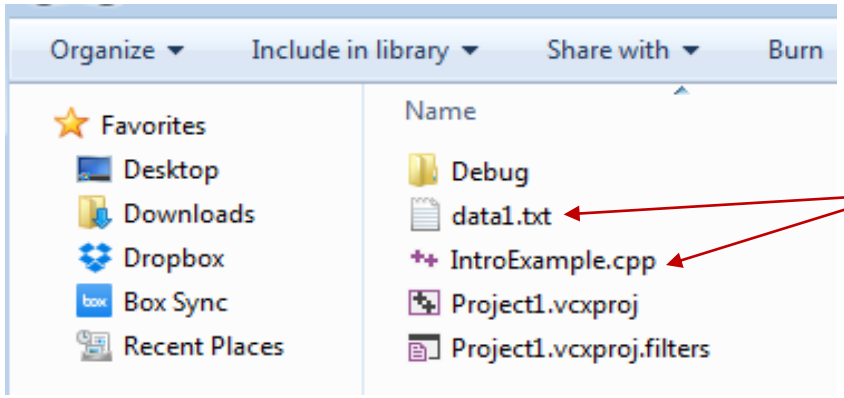
For example:

- Create a program that:
  - Defines integers i (counter variable) and y (given a value later in the code)
  - Prints the address of the file to the screen
  - Starting at i=1 and for a total of 3 times, calculates y=2*i
  - Prints i and y to the screen and to a file called data1.txt

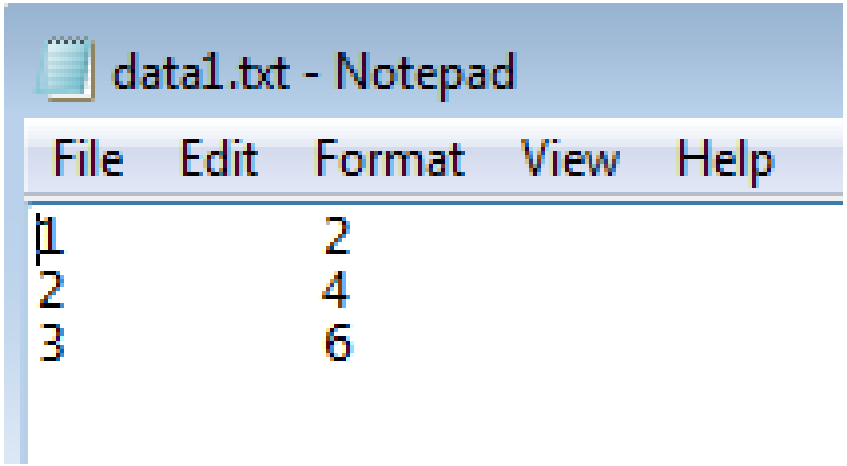
For example:

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

int main()
{
    int i, y, z;

    //Create file pointer
    FILE *outfile;

    //Open file
    outfile = fopen("data1.txt", "w"); //"w" because we want to write in it
    //File can be .txt or .dat

    //Print address of file (pointer is address of file - we need %p)
    printf("Address of file is %p\n\n", outfile);

    //Create loop to print i and y to screen and to file
    for (i = 1; i <= 3; i++)
    {
        y = 2 * i;

        //Print to screen
        printf("%d \t %d\n\n", i, y);
        //Print to file
        fprintf(outfile, "%d \t %d\n", i, y);
    }

    printf("\n");
    fclose(outfile);


}
```

```
Address of file is 002ACB78

1         2

2         4

3         6

Press any key to continue . . .
```



File saved in same location as .cpp file



Example, continued:

- Modify your program so that it also does the following:
  - Creates int variable z = y*4
  - Prints value of z in screen and in same data1.txt file

Example, continued:

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

int main()
{
    int i, y, z;

    //Create file pointer
    FILE *outfile;

    //Open file
    outfile = fopen("data1.txt", "w"); //"w" because we want to write in it
    //File can be .txt or .dat

    //Print address of file (pointer is address of file - we need %p)
    printf("Address of file is %p\n\n", outfile);

    //Create loop to print i and y to screen and to file
    for (i = 1; i <= 3; i++)
    {
        y = 2 * i;

        //Print to screen
        printf("%d \t %d\n\n", i, y);
        //Print to file
        fprintf(outfile, "%d \t %d\n", i, y);
    }

    printf("\n");
    fclose(outfile);

    //Re-open file to add new
    outfile = fopen("data1.txt", "a"); //"a" because we want to add to it

    //New calculation
    z = y * 4;

    //Print to screen
    printf("%d\n\n", z);
    //Print to file
    fprintf(outfile, "%d\n", z);

    //Close file
    fclose(outfile);

}
```

```
Address of file is 006ACB78

1         2

2         4

3         6


24

Press any key to continue . . .
```

data1.txt - Notepad

File Edit Format View Help

```
1         2
2         4
3         6
24
```

We can open and close files in order to read or write information, as follows:

- Use fscanf to read data from a file that you have opened:

```
fscanf(pointer_variable_name,“data types”, &variable_names);
```

Notice this is the same syntax as scanf, but we add the pointer variable in the beginning

- Remember to first define a pointer with the FILE command – you need to do this before you can open the file and read data!

For example:

- Create a program that:
  - Defines integers i (counter variable) and y (given a value later in the code)
  - Creates a file to print to and prints the address of the file to the screen
  - Starting at i=1 and for a total of 3 times, calculates y=4*i
  - Prints i and y to the screen and to a file called data2.txt
  - Closes file, then re-opens it, reads the data, and assigns it to variables Var1 and Var2
  - Prints the data read from the file to the screen
  - Prints the address of the file to the screen

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

int main()
{
    int i, y, Var1, Var2;

    //Pointer for file to which I am printing
    FILE *outfile;
    //Pointer for file from which I am reading
    FILE *infile;

    //Open file where I am printing
    outfile = fopen("data2.txt", "w");

    //Print address of file to screen
    printf("Address of file is %p\n\n", outfile);

    //Print data to screen and file
    printf("Data printed to screen:\n");
    for (i = 1; i <= 3; i++)
    {
        y = 4 * i;
        printf("%d \t %d \n", i, y);
        fprintf(outfile, "%d \t %d \n", i, y);
    }

    printf("\n");

    //Close file
    fclose(outfile);

    //Open file to read the data
    infile = fopen("data2.txt", "r");  //"r" because we are reading from the file

    //Print file address to screen
    printf("Address of file is %p\n\n", infile);

    //Read data from file and print to screen
    printf("Data read from file is:\n");
    for (i = 1; i <= 3; i++)
    {
        fscanf(infile, "%d %d", &Var1, &Var2);  //Remember the & to read data
        printf("%d \t %d\n", Var1, Var2);
    }

    //Close file
    fclose(infile);

}
```

```
Address of file is 0068CB78

Data printed to screen:
1          4
2          8
3          12

Address of file is 0068CB78

Data read from file is:
1          4
2          8
3          12
Press any key to continue . . .
```

Same address – it is the same file!

data2.txt - Notepad

File Edit Format View Help

```
1          4
2          8
3          12
```

Practice Problem

- Write a program that:
  - Defines an integer i that controls a for loop for 5 iterations
  - Defines a float y that is 5*i
  - Writes to a file using the pointer name “WriteToFile”
  - Prints the address of the file (the pointer) to the screen
  - Saves the file name as “WriteF.txt”
  - For each loop iteration, print the values of i and y to the file (y should have 3 decimal places)
  - For each loop iteration, print the values of i and y to the screen (y should have 3 decimal places)

Open the data file and compare to your output screen!

Submit .cpp file called “Mena_Time_fprintfTeam#” (“Mena_10am_fprintfL01”) into Classwork folder

Practice Problem

- Create a file (“data3.txt”). The file should have data as follows:

  3
  6
  9
  12

- Create a program that:
  - Reads data from “data3.txt” (use a for loop) and prints the data to the screen
  - Creates a variable y=2*user_num, where user_num is a value provided by the user
  - Prints “The value for y is (value)” to a file called “Print1.txt” and also to the screen
  - Your output screen should look like this:

```
Data read from file is:
3
6
9
12
Please provide a number:
15
The value for y is 30
Press any key to continue . . .
```

Submit .cpp file called
“Mena_Time_fscanfTeam#”
(“Mena_10am_fscanfL01”)

Print1.txt - Notepad

File Edit Format View Help

```
The  value for y is 30
```

To read an array of numbers from a file, put the fscanf in a loop and use array notation

For example, use this:

```
n=10;
for (i=0; i<=n-1; i++)
{
        fscanf(infile,"%lf %lf",&x[i],&y[i]);
}
```

…to read this file where x is the first column of numbers and y is the second:

1.0	2.3
2.0	3.2
3.0	4.5
4.5	3.5
5.6	3.4
2.3	5.6
4.5	5.6
3.4	6.7
4.5	7.8
5.6	5.6

We can check for bad file names, knowing that if a file does not exist, the pointer variable will be set to NULL (00000000)

- So we can check for this in our code:

```
prior commands
if (infile == NULL)
        {
                printf("Bad data file \n\n");
        }
other commands
```

When we don’t know how many data points are in our file (how many times to use fscanf), we can use EOF

- “end of file”

To use EOF, we can use scanf and fscanf to get the number of items read

- We can assign a variable to the input of the fscanf:

This variable will be equal to the number of inputs being read, so 2

```
status= fscanf(infile,"%d %lf",&x, &y);
```

This does NOT have the data values (the data values are assigned to x and y), but rather the number of values being read

- Then, we can compare that value to the EOF character:

```
if (status==EOF) or if (status != 2)
break;
```

When the end of the file is reached and no data has been read, the output is -1, associated with the EOF

For example:

- Create a file “WriteF.txt” that has this data:

5
10
15
20
25

- Write a program that:
  - Opens “WriteF.txt”, reads the data, stores it in array x
  - Uses EOF to read the data
  - Prints the values of array x to the screen
  - Prints the location of “WriteF.txt”
  - Output should look like this:

```
Address of file is 0049CB78
Here's the data that is read from the file:
 5
 10
 15
 20
 25
Press any key to continue . . .
```

For example:

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

int main()
{
    //Declare variables
    int x[50], status = 1, i = 0;

    //Pointer for file from which I am reading
    FILE *infile;

    //Open file
    infile = fopen("WriteF.txt", "r");

    //Print location of file
    printf("Address of file is %p\n\n", infile);

    //Read data from file
    printf("Here's the data that is read from the file: \n\n");
    while (status != EOF)
    {
        status = fscanf(infile, "%d", &x[i]);  //Check that there is data to be read
        if (status == EOF)
            break;
        printf(" %d \n\n", x[i]);
        i = i + 1;
    }

    fclose(infile);

}
```

Example, continued

- Modify your program to print the total number of points that were read from the file

- Modify your program to print the total number of points that were read from the file, and use the status variable to calculate the total number

Example, continued

```c
int main()
{
    //Declare variables
    int x[50], status = 1, i = 0, numLoops=0;

    //Pointer for file from which I am reading
    FILE *infile;

    //Open file
    infile = fopen("WriteF.txt", "r");

    //Print location of file
    printf("Address of file is %p\n\n", infile);

    //Read data from file
    printf("Here's the data that is read from the file: \n\n");
    while (status != EOF)
    {
        status = fscanf(infile, "%d", &x[i]);  //Check that there is data to be read
        printf("Status= %d\n", status); //Print the value of status variable

        if (status == EOF)
            break;
        printf(" %d \n\n", x[i]);
        i = i + 1;
        numLoops = numLoops + status;
    }

    printf("Total number of points= %d\n\n", i);
    printf("Total number of points (using status variable)= %d\n\n", numLoops);

    fclose(infile);

}
```

```
Address of file is 004ECB78

Here's the data that is read from the file:

Status= 1
 5

Status= 1
 10

Status= 1
 15

Status= 1
 20

Status= 1
 25

Status= -1
Total number of points= 5

Total number of points (using status variable)= 5

Press any key to continue . . .
```

You can also use fscanf to check for bad data

- Determine the number of items read and break if it is not equal to a particular value

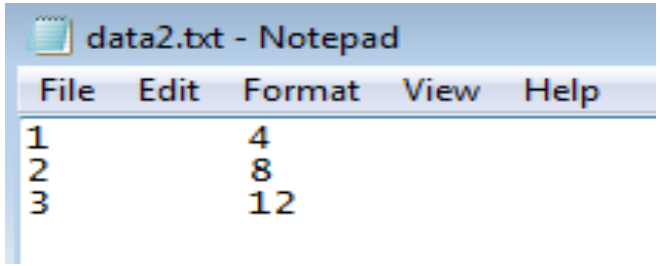
We can allow the user to enter the file they want to open

- To read in the name of a file typed by the user from the keyboard, a string variable (char array) must first be declared

- Remember to use %s to read strings

For example:

- Create file “data2.txt”

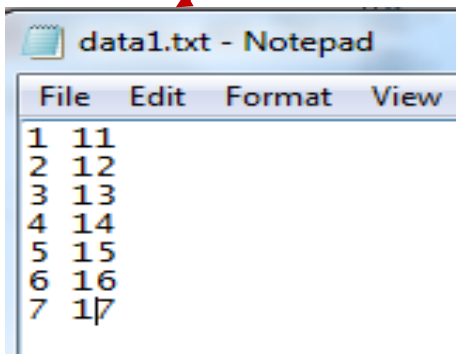


- Create a program that will ask the user to enter a file name and check if the file exists, save data to integer variable x and double variable y, and print x and y to the screen (print only 2 decimal spaces) each time they are assigned a value
- The program should check for incomplete data

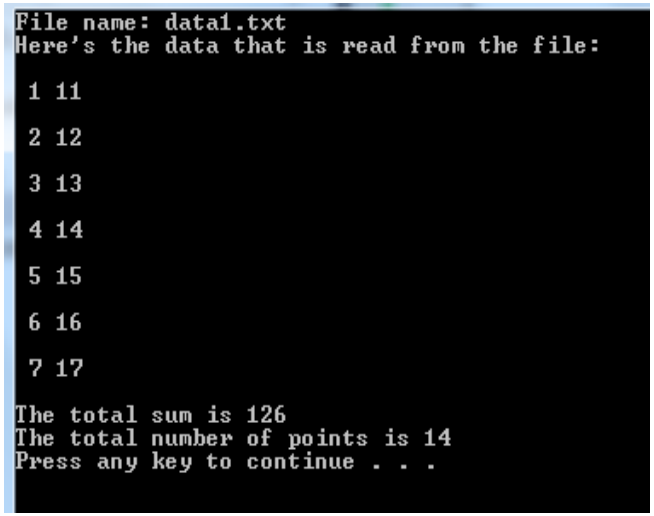
For example:

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

int main()
{
    int x, status = 2; //status=2, meaning there should be two columns
    double y;
    char filename[50]; //Declare the file name as a string

    //Create pointer
    FILE *infile;

    do
    {
        //Ask for file name
        printf("File name: \n");
        scanf("%s", filename);
        //Open file
        infile = fopen(filename, "r");
    } while (infile == NULL);

    while (status == 2 && status != EOF)
    {
        status = fscanf(infile, "%d %lf", &x, &y);
        printf("Number of values read was %d\n", status);
        if (status == EOF)
        {
            break;
        }
        if (status != 2 && status != EOF)
        {
            printf("Incomplete data!");
            break;
        }
        printf("%d %.2lf \n", x, y);
    }
        //Close file
        fclose(infile);
}
```

Practice Problem



Output should look like this



- Create the file "data1.txt", with the data in columns (x y)
- Ask the user for the file name, and check for user error (Hint: use NULL)
- Read the x and y variables as arrays from the file
  - Both x and y should be in int format, so declare them like this: int x[10], y[10] → 10 would then be the maximum number of pairs that could be read in
  - Use EOF to find the end of the file
- Write the x and y pairs to the screen and to another data file called "OutputData.txt"
- Get the sum of all the values and print "The sum is: *sum*" to the screen and to the "OutputData.txt" file
- Get the total number of points in the file and print "The total number of points is: *number*" to the screen and to the "OutputData.txt" file

Submit .cpp file called
"Mena_Time_CPracticeTeam#"
("Mena_10am_CPracticeL01")