ENGR 0012 – Engineering Problem Solving

Goals for this week:

- Create functions in C++
- Compare global and local variables

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

Some things to remember when reading from a file:

- When using fopen, fscanf and fclose, use the pointer variable, NOT the data file name
- When reading a data file with more than one column, consider comparing the fscanf output to the number of columns – do this within the while loop
- Initialize the while loop variables to zero (not 1), and increment them in the loop
- Make sure data files are stored in the right place (same directory that your C code is in)

As in MATLAB, we can use modular design in C (functions!)

- BUT:
- C functions can only return one variable back to the main (although more than one can be sent to the function) – note that this is different for arrays and pointers
- C functions are included in the same script (file), not written in separate files as with MATLAB m-files

To use functions in C, we need the following lines of code:

1. Prototype definition before the main

2. Function call from the main

3. Function declaration statement after the main (where the code for the function begins in your script)

To use functions in C, we need the following lines of code:

1. Prototype definition before the main

One of four types:

1. syntax: void function_name(void);
2. syntax: void function_name (list of variable types to be sent, separated by commas);
3. syntax: variable_type function_name(void);
4. syntax: variable_type function_name(list of variable types to be sent, separated by commas);

To use functions in C, we need the following lines of code:

2. Function call from the main

One of four types:

1. syntax: function_name();
2. syntax: function_name(list of variables separated by commas);
3. syntax: variable_name= function_name();
4. syntax: variable_name=function_name(list of variables separated by commas);

To use functions in C, we need the following lines of code:

3. Function declaration statement after the main (where the code for the function begins in your script)
    - After the closing } of the main – indicates the beginning of the commands in the function
    - No semicolon at the end
    - Commands that follow are enclosed in {}

    - Note: if a variable is being returned to the main, we need to include this line of code within the function: return(VarName)

Example 1

- Create a program that calls a function to display your name, date, and class

Example 1

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

void userinfo(void);  //Function prototype, before the main

int main(void)
{
    userinfo();  //Function call, within the main


}

void userinfo(void)  //Function definition, after the main
{
    printf("\nYour name & group number");
    printf("\nThe date");
    printf("\nDescription of output");
}
```

```
Your name & group number
The date
Description of outputPress any key to continue . . .
```

Example 2

- Create a program that:
  - Declares an integer m=15 and a float y=30.5
  - Prints the values to the screen
  - Calls function2, sending m and y to the function (does not receive anything back)
  - Print the values of m and y to the screen
  - function2 should:
    - Receive inputs m and y
    - Assign m=3 and y=50
    - Print the values from the function

Example 2

```
The values are m=15 and y=30.500000

The values from the function are m=3 and y=50.000000

The values after the function are m=15 and y=30.500000

Press any key to continue . . .
```

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

 void function2(int, double);  //Need variable types, names are optional

int main(void)
{
    //Declare variables
    int m = 15;
    double y = 30.5;

    printf("The values are m=%d and y=%lf\n\n", m, y);
    function2(m, y);  //Need variable names, not types
    printf("The values after the function are m=%d and y=%lf\n\n", m, y);

}


void function2(int m, double y)  //Need variable types and variable names
{
    m = 3;
    y = 50;
    printf("The values from the function are m=%d and y=%lf\n\n", m, y);
}
```

Example 3

- Create a program that:
  - Declares an integer m=15 and a float y=30.5
  - Calls function2, sending m and y to the function, and should receive Result back
  - function2 should:
    - Receive inputs m and y, calling them p and q
    - creates double variable z in the function, z=p+q
    - Sends z to the main
  - Main prints Result to the screen

Example 3

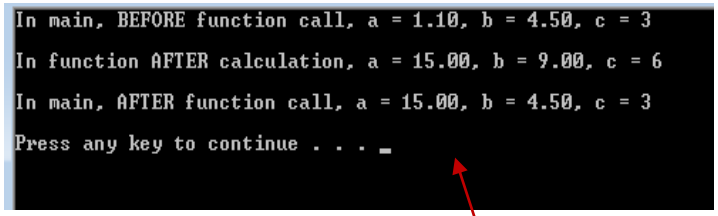
```
Result = 45.500000

Press any key to continue . . .
```

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

double function2(int, double);  //Need variable types

int main(void)
{
    //Declare variables
    int m = 15;
    double y = 30.5, Result;

    Result = function2(m, y);  //Need variable names

    //Print Result
    printf("Result = %lf\n\n", Result);


}


double function2(int p, double q)  //Need variable types and names of vars coming in
{
    double z;
    z = p + q;

    //Need to indicate which variable is being returned
    return(z);
}
```

Some things to remember regarding functions:

- Variable names can differ in the function call and function declaration
- The order of the variables within the statements is important
- Data type of the variables passed to and from the function must be consistent with type declared
- Only the value of the variable being returned from the function is changed in the main (except in the case of arrays or pointer variables)

Practice Problem



Does it make sense why these are the results?

- Create a program that:
  - Creates double variables a=1.1 and b=4.5
  - Creates integer variable c=3
  - Prints a, b, and c to the screen (2 decimal places, if applicable)
  - Calls my_function, sending b and c, and receiving a
  - Prints a, b, and c to the screen (2 decimal places, if applicable)

- my_function should:
  - Create double variable a
  - Assign b=b*2, c=c*2, a=b+c
  - Prints a, b, and c from the function (2 decimal places, if applicable)
  - Returns variable a to the main

Submit .cpp file called “Mena_Time_Prob1Team#”
(“Mena_10am_Prob1L01”) into Classwork folder

Practice Problem

$$A = P\left[\frac{i(1+i)^N}{(1+i)^N - 1}\right]$$

- func_header should not send any variables or return any. It should print a header with your names and group number.
- In the main program, have the user enter a value for a principle amount of money to be borrowed (P), the number of months to pay the loan back (N), and a monthly interest rate (i). The user should enter the value as a percent and the program should divide it by 100 for the calculation.
- func_A should receive the three values entered above and return A. The main should then print out the value of A.
- Put everything (except func_header) in a loop so the user can try several payment plans.

Try your program with these values:

P=$3000, N=30, i=1%

P=$100,000, N=360, i=.5%

P=$20,000, N=48, i=.2%

Answers you should get are:

116.24; 599.55; 437.40

Submit .cpp file called "Mena_Time_Prob2Team#" ("Mena_10am_Prob2L01") into Classwork folder

Global versus local variables

```
(Global Scope)
#include <stdio.h>
#include<stdlib.h>
// #define GlobalVar 4.5

double GlobalVar = 4.5;

double function2(double);


int main(void)
{
    double y = 2, NewVar;

    //Print GlobalVar
    printf("\nThe value of GlobalVar in main is GlobalVar = %lf\n\n", GlobalVar);

    //Only y is sent; GlobalVar not sent
    NewVar = function2(y);

    //Print GlobalVar after function
    printf("\nThe value of GlobalVar AFTER the function is  GlobalVar = %lf\n", GlobalVar);


}

double function2(double x)
{
    double k;

    k = x*GlobalVar;

    printf("\nThe value of GlobalVar in function2 is GlobalVar = %lf\n\n", GlobalVar);

    GlobalVar = 3;

    return k;  //Only k returned; GlobalVar not returned
}
```

Global variable

Local variables

Global variable

Local variables

Global variable

```
The value of GlobalVar in main is GlobalVar = 4.500000


The value of GlobalVar in function2 is GlobalVar = 4.500000


The value of GlobalVar AFTER the function is  GlobalVar = 3.000000
Press any key to continue . . .
```

Functions in C can only return one variable to the main

- What if we need to return multiple variables?
- Consider using arrays

With arrays, we can pass multiple values to and from a function

- When arrays are passed to a function, only the starting location is actually passed

- If values in these arrays are changed within a function, they will be changed in the main as well (not true for other types of variables!)

Notice the notation:

- For prototype: Dimension in a one dimensional array and first dimension in multiple dimension arrays should be blank
- For function call: Do not include the brackets
- For function declaration: Dimension in a one-dimensional array and first dimension in two-dimensional arrays should be blank

For example:

For example:

Dimension in 1-D array and first dimension in 2-D array are blank

```
Here are the arrays BEFORE the function call:
aa = [4  5]
bb = [5 -3 ; 1 2]
Here are the arrays AFTER the function call:
aa = [14  15]
bb = [15 7 ; 11 12]
Press any key to continue . . .
```

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

void my_function(int[], int[][2]);  //Notice which [] are empty


int main()
{
    //Declare arrays
    int aa[3] = { 4, 5 };
    int bb[2][2] = { { 5, -3},{ 1, 2} };

    printf("Here are the arrays BEFORE the function call:\n\naa = [%d  %d] \n\nbb = [%d %d ; %d %d]\n\n", aa[0], aa[1], bb[0][0], bb[0][1], bb[1][0], bb[1][1]);

    my_function(aa, bb);

    //Notice values of arrays after the function
    printf("Here are the arrays AFTER the function call:\n\naa = [%d  %d] \n\nbb = [%d %d ; %d %d]\n\n", aa[0], aa[1], bb[0][0], bb[0][1], bb[1][0], bb[1][1]);


}

void my_function(int aa[], int bb[][2])  //Notice which [] are empty
{
    int i;

    //Change values in arrays
    for (i = 0; i <= 1; i++)
    {
        aa[i] = aa[i] + 10;
        bb[0][i] = bb[0][i] + 10;
        bb[1][i] = bb[1][i] + 10;
    }
}
```

No brackets

Dimension in 1-D array and first dimension in 2-D array are blank

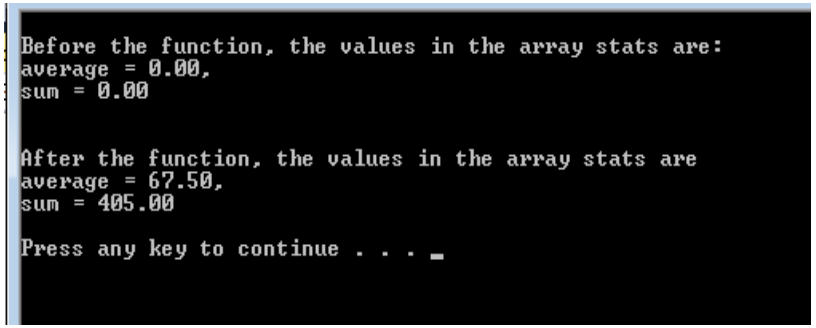
Notice there is no return – why?

Practice Problem

- Create a program that:
- Asks the user to input 2 integer values
- Stores these values in an array
- Uses a function to calculate the product and sum of the two integers (these should be stored in another array)
- Displays the product and sum in the main program (print results to screen)
- Asks the user if they want to repeat, using a do-while loop

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

//Function
void function1(int[], int[]);

int main(void)
{
    //Declare variables
    int xyarray[2], solutionArray[2];
    char again = 'y';

    //Do-while loop
    do
    {
        //Ask for user input and store in array
        printf("\nEnter one integer:");
        scanf("%d", &xyarray[0]);

        printf("\nEnter another integer:");
        scanf("%d", &xyarray[1]);

        //Function call
        function1(xyarray, solutionArray);

        //Print values
        printf("\nThe product is %d", solutionArray[0]);
        printf("\nThe sum is %d", solutionArray[1]);

        //Do again?
        printf("\nAgain?");
        scanf(" %c", &again);
    } while (again == 'y' || again == 'Y');


}

void function1(int xyarray[], int solutionArray[])
{
    int prod1, sum1;
    prod1 = xyarray[0] * xyarray[1];
    sum1 = xyarray[0] + xyarray[1];
    solutionArray[0] = prod1;
    solutionArray[1] = sum1;
}
```

Practice Problem



- Declare two arrays:
- An array of test scores:

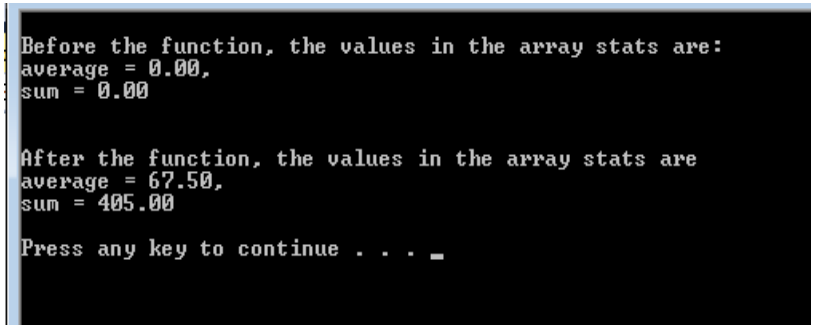
  scores=[70,80,60,45,95,55]
- An empty array that will store the average and sum values calculated in a function

  stats=[0,0]
- A function will calculate the average and the sum of these values
- From the main, print the average and sum (the values in the stats array) before the function
- After the function, print the average and sum from the main

Submit .cpp file called “Mena_Time_StatsTeam#”
(“Mena_10am_StatsL01”)

From slides by N. Vidic

Practice Problem - Modify



- Declare two arrays:
- An array of test scores:

  scores=[70,80,60,45,95,55]

  Read this data from a file and use EOF!
- An empty array that will store the average and sum values calculated in a function

  stats=[0,0]
- A function will calculate the average and the sum of these values
- From the main, print the average and sum (the values in the stats array) before the function
- After the function, print the average and sum from the main

Review: What is the output?

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

int main()
{
    //Declare variables
    int a = 3, b = 6, c = 9, d = 12, e, f;
    double alfa = 2, beta = 3, gamma = 3.3, delta, epsilon;

    //Math operations
    delta = alfa*(a*b / c);
    e = (d%c)*b;

    epsilon = (alfa*alfa / beta)*(c / b);
    f = alfa*gamma;

    //Print results
    printf("Hello!");
    printf("\ndelta=%5.1lf \ne=%7d", delta, e);
    printf("\nepsilon=%lf \nf=%d", epsilon, f);
    printf("\nalfa=%4.2lf", alfa);

    printf("\n");


}
```

Review: What is the output?

```
Hello!
delta=  4.0
e=     18
epsilon=1.333333
f=6
alfa=2.00
Press any key to continue . . .
```

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

int main()
{
    //Declare variables
    int a = 3, b = 6, c = 9, d = 12, e, f;
    double alfa = 2, beta = 3, gamma = 3.3, delta, epsilon;

    //Math operations
    delta = alfa*(a*b / c);
    e = (d%c)*b;

    epsilon = (alfa*alfa / beta)*(c / b);
    f = alfa*gamma;

    //Print results
    printf("Hello!");
    printf("\ndelta=%5.1lf \ne=%7d", delta, e);
    printf("\nepsilon=%lf \nf=%d", epsilon, f);
    printf("\nalfa=%4.2lf", alfa);

    printf("\n");


}
```

Review: What is the output?

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

void main(void)
{
    int i = 4, j = 2;
    char A1[40] = "Today is a good day";
    char A2[40] = "I love Engineering";
    char A3[40] = "Hi";
    char NEW[30] = { A1[i],A2[i + 1],A1[i + 3] };

    printf("%s\n", A3);
    printf("%s\n", NEW);
    for (i = 6; i<15; i++)
    {
        printf("%c\n", A2[i]);
    }

}
```

Review: What is the output?

```
Hi
yes

E
n
g
i
n
e
e
r
Press any key to continue . . .
```

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

void main(void)
{
    int i = 4, j = 2;
    char A1[40] = "Today is a good day";
    char A2[40] = "I love Engineering";
    char A3[40] = "Hi";
    char NEW[30] = { A1[i],A2[i + 1],A1[i + 3] };

    printf("%s\n", A3);
    printf("%s\n", NEW);
    for (i = 6; i<15; i++)
    {
        printf("%c\n", A2[i]);
    }

}
```