ENGR 0012 – Engineering Problem Solving

Goals for this week:

- Create arrays
- Perform math operations
- Use looping and branching in C++

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

Practice Problem
What is the output? Why?

(a)
(default)

```c
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
    float a = 3.2;
    int b = 5;
    double c = 28.4;

    printf("\n%f\n", a);
    printf("\n%d\n", b);
    printf("\n%lf\n", c);

    system("pause");
}
```

3.200000
5
28.400000

(b)

```c
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
    float a = 3.2;
    int b = 5;
    double c = 28.4;

    printf("\n%d\n", b/2);

    printf("\n%8.2f\n", a);
    printf("\n%-3.1lf\n", c);

    system("pause");
}
```

2
_ _ _ _3.20
28.4

We can declare string arrays in C

- If it's a single character → char Var1='a'
- If it's a string → char Var2[20]

Length of the string array

- Add elements one by one, with the last one being \0

Instead of adding elements one by one, use the strcpy command in the string.h library

- Use this format:

strcpy(Var1, Varb)

First variable is the one you are assigning a value to

Varb is copied to Var1

strcpy(Var2, “My string array”)

Var2 is now “My string array”

To use printf for a string array, use %s

```
printf("Here is Var2: %s",Var2)
```

For example:

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

void main()
{
    char Var1 = 'a';
    char Var2[25];

    strcpy(Var2, "How are you?");

    //Use printf to print
    printf("\nThe letter is:%c", Var1);
    printf("\nThe string array is:\%s\n", Var2);

}
```

```
The letter is:a
The string array is:How are you?
```

Practice Problem

- Define a one-dimensional array
- Define a two-dimensional array
- Print the first element and last element of each of your arrays
- Define a string variable equal to “Programming in C” and print it to the screen

C allows mathematical operations

- Addition +
- Subtraction –
- Multiplication *
- Division /
- Remainder (for integers only) %

You can perform the operation in the program or within printf

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

void main()
{
    int A = 3, B = 7, C;
    float D = 3.0, E = 7.0, F;

    //Mathematical operations in program
    C = A + B;
    F = D + E;

    //Mathematical operations within printf
    printf("\nAdding these numbers C = %3d, F = %5.2f", C, F);
    printf("\nAdding these numbers C = %3d, F = %5.2f", A + B, D + E);
    printf("\n");

}
```

```
Adding these numbers C =  10, F = 10.00
Adding these numbers C =  10, F = 10.00
Press any key to continue . . .
```

What is the difference between division and remainder? (Note this is for integers - % is undefined for float and double)

(for int A=3, B=7)

- B/A will give 2
- B%A will give 1 (3 goes into 7 2 times with a remainder = 1)

- A/B will give 0

C allows mathematical functions – see your textbook to learn more!

- Remember to include the math.h library

- pow (not ^) for exponents
  - pow(x,y) gives x to the power of y

C allows mathematical functions – see your textbook to learn more!

- Numbers must be in radians for trig functions
  - So multiply degrees by PI/180 before using these functions
  - But remember that PI must be defined by the user

- Natural log is log and base 10 log is log10

What happens when different data types are used in mathematical operations?

- Two integers produce an integer
- Two reals produce a real
- Mix of reals and integers produces a real, but it will be stored according to how it was declared
- Numbers in arithmetic operations typed without a decimal point will be treated as integers, but it will be stored according to how it was declared

What happens when different data types are used in mathematical operations?

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

void main()
{
    int IntA = 15, IntB = 2, IntC, IntD, IntE;
    double DoubleA = 15, DoubleB = 2, DoubleC, DoubleD, DoubleE;

    //Operations on integer variables
    IntC = IntA / IntB;
    IntD = 15 / 2;

    IntE = 15 / 2.0;  //real, but IntE is an int

    //Operations on double variables
    DoubleC = DoubleA / DoubleB;
    DoubleD = 15 / 2; //integeres, but DoubleD is a double
    DoubleE = 15 / 2.0;

    printf("\nThe integer variables are IntC=%d, IntD=%d, and IntE=%d\n", IntC, IntD, IntD);
    printf("\nThe double variables are DoubleC=%lf, DoubleD=%lf, DoubleE=%lf\n", DoubleC, DoubleD, DoubleE);



}
```

```
The integer variables are IntC=7, IntD=7, and IntE=7

The double variables are DoubleC=7.500000, DoubleD=7.000000, DoubleE=7.500000
Press any key to continue . . .
```

What happens when different data types are used in mathematical operations?

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

void main()
{
    int IntA = 15, IntB = 2, IntC, IntD, IntE;
    double DoubleA = 15, DoubleB = 2, DoubleC, DoubleD, DoubleE;

    //Operations on integer variables
    IntC = IntA / IntB;
    IntD = 15 / 2;

    IntE = 15 / 2.0;  //real, but IntE is an int

    //Operations on double variables
    DoubleC = DoubleA / DoubleB;
    DoubleD = 15 / 2; //integeres, but DoubleD is a double
    DoubleE = 15 / 2.0;

    printf("\nThe integer variables are IntC=%d, IntD=%d, and IntE=%d\n", IntC, IntD, IntD);
    printf("\nThe double variables are DoubleC=%lf, DoubleD=%lf, DoubleE=%lf\n", DoubleC, DoubleD, DoubleE);

}
```

15/2 = 7

```
The integer variables are IntC=7, IntD=7, and IntE=7

The double variables are DoubleC=7.500000, DoubleD=7.000000, DoubleE=7.500000
Press any key to continue . . .
```

What happens when different data types are used in mathematical operations?

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

void main()
{
    int IntA = 15, IntB = 2, IntC, IntD, IntE;
    double DoubleA = 15, DoubleB = 2, DoubleC, DoubleD, DoubleE;

    //Operations on integer variables
    IntC = IntA / IntB;
    IntD = 15 / 2;

    IntE = 15 / 2.0;  //real, but IntE is an int

    //Operations on double variables
    DoubleC = DoubleA / DoubleB;
    DoubleD = 15 / 2; //integeres, but DoubleD is a double
    DoubleE = 15 / 2.0;

    printf("\nThe integer variables are IntC=%d, IntD=%d, and IntE=%d\n", IntC, IntD, IntD);
    printf("\nThe double variables are DoubleC=%lf, DoubleD=%lf, DoubleE=%lf\n", DoubleC, DoubleD, DoubleE);

}
```

15/2 = 7

```
The integer variables are IntC=7, IntD=7, and IntE=7

The double variables are DoubleC=7.500000, DoubleD=7.000000, DoubleE=7.500000
Press any key to continue . . .
```

What happens when different data types are used in mathematical operations?

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

void main()
{
    int IntA = 15, IntB = 2, IntC, IntD, IntE;
    double DoubleA = 15, DoubleB = 2, DoubleC, DoubleD, DoubleE;

    //Operations on integer variables
    IntC = IntA / IntB;
    IntD = 15 / 2;

    IntE = 15 / 2.0;  //real, but IntE is an int

    //Operations on double variables
    DoubleC = DoubleA / DoubleB;
    DoubleD = 15 / 2; //integeres, but DoubleD is a double
    DoubleE = 15 / 2.0;

    printf("\nThe integer variables are IntC=%d, IntD=%d, and IntE=%d\n", IntC, IntD, IntD);
    printf("\nThe double variables are DoubleC=%lf, DoubleD=%lf, DoubleE=%lf\n", DoubleC, DoubleD, DoubleE);

}
```

```
The integer variables are IntC=7, IntD=7, and IntE=7

The double variables are DoubleC=7.500000, DoubleD=7.000000, DoubleE=7.500000
Press any key to continue . . .
```

15/2.0 = 7.5

IntE=7

What happens when different data types are used in mathematical operations?

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

void main()
{
    int IntA = 15, IntB = 2, IntC, IntD, IntE;
    double DoubleA = 15, DoubleB = 2, DoubleC, DoubleD, DoubleE;

    //Operations on integer variables
    IntC = IntA / IntB;
    IntD = 15 / 2;

    IntE = 15 / 2.0;  //real, but IntE is an int

    //Operations on double variables
    DoubleC = DoubleA / DoubleB;
    DoubleD = 15 / 2; //integeres, but DoubleD is a double
    DoubleE = 15 / 2.0;

    printf("\nThe integer variables are IntC=%d, IntD=%d, and IntE=%d\n", IntC, IntD, IntD);
    printf("\nThe double variables are DoubleC=%lf, DoubleD=%lf, DoubleE=%lf\n", DoubleC, DoubleD, DoubleE);

}
```

15/2 = 7.5

```
The integer variables are IntC=7, IntD=7, and IntE=7

The double variables are DoubleC=7.500000, DoubleD=7.000000, DoubleE=7.500000
Press any key to continue . . .
```

What happens when different data types are used in mathematical operations?

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

void main()
{
    int IntA = 15, IntB = 2, IntC, IntD, IntE;
    double DoubleA = 15, DoubleB = 2, DoubleC, DoubleD, DoubleE;

    //Operations on integer variables
    IntC = IntA / IntB;
    IntD = 15 / 2;

    IntE = 15 / 2.0;  //real, but IntE is an int

    //Operations on double variables
    DoubleC = DoubleA / DoubleB;
    DoubleD = 15 / 2; //integeres, but DoubleD is a double
    DoubleE = 15 / 2.0;

    printf("\nThe integer variables are IntC=%d, IntD=%d, and IntE=%d\n", IntC, IntD, IntD);
    printf("\nThe double variables are DoubleC=%lf, DoubleD=%lf, DoubleE=%lf\n", DoubleC, DoubleD, DoubleE);

}
```

15/2 = 7

```
The integer variables are IntC=7, IntD=7, and IntE=7

The double variables are DoubleC=7.500000, DoubleD=7.000000, DoubleE=7.500000
Press any key to continue . . .
```

DoubleD=7.000000

What happens when different data types are used in mathematical operations?

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

void main()
{
    int IntA = 15, IntB = 2, IntC, IntD, IntE;
    double DoubleA = 15, DoubleB = 2, DoubleC, DoubleD, DoubleE;

    //Operations on integer variables
    IntC = IntA / IntB;
    IntD = 15 / 2;

    IntE = 15 / 2.0;  //real, but IntE is an int

    //Operations on double variables
    DoubleC = DoubleA / DoubleB;
    DoubleD = 15 / 2; //integeres, but DoubleD is a double
    DoubleE = 15 / 2.0;

    printf("\nThe integer variables are IntC=%d, IntD=%d, and IntE=%d\n", IntC, IntD, IntD);
    printf("\nThe double variables are DoubleC=%lf, DoubleD=%lf, DoubleE=%lf\n", DoubleC, DoubleD, DoubleE);

}
```

15/2.0 = 7.5

```
The integer variables are IntC=7, IntD=7, and IntE=7

The double variables are DoubleC=7.500000, DoubleD=7.000000, DoubleE=7.500000
Press any key to continue . . .
```

Practice Problem
What is the output? Why?

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

void main()
{
    float a = 11, c;
    int b = 3, d, e;

    printf("%f\n", a);
    printf("%d\n", b);

    c = a / b;
    d = a / b;

    e = b / 2.0;

    printf("The value of c is %4.2f\n", c);
    printf("The value of d is %05d\n", d);
    printf("The value of e is %d\n", e);


}
```

How many total characters?
How many after the decimal?

Practice Problem
What is the output? Why?

```
11.000000
3
The value of c is 3.67
The value of d is 00003
The value of e is 1

Press any key to continue . . .
```

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

void main()
{
    float a = 11, c;
    int b = 3, d, e;

    printf("%f\n", a);
    printf("%d\n", b);

    c = a / b;
    d = a / b;

    e = b / 2.0;

    printf("The value of c is %4.2f\n", c);
    printf("The value of d is %05d\n", d);
    printf("The value of e is %d\n", e);


}
```

How many total characters?
How many after the decimal?

What happens when different data types are used in mathematical operations?

- A cast operator can be used to assign a data type to a variable

- Thus if C is to be a real: C= (double)A/B
- or if it is to be an integer: C= A/(int)B

Practice Problem

- Write a short program that defines four variables: two integers (a and b) and two doubles (c and d)
  - Assign values of a and c within the program
  - Ask the user to provide values of b and d (use scanf)
- Create a variable that will be equal to a+c, one equal to a+b, and one equal to d to the power of 2 (include <math.h> library)
  - Note: You'll end up defining additional variables, so be sure to declare them at the beginning of your program! Should they be integers or doubles?
- Print out all the integer variables and all the double variables to the screen
- Include comments!

Submit .cpp file called "Mena_Time_CMathGr#" ("Mena_10am_CMathL01") into Classwork folder

C has some helpful assignment operators that we can use as shortcuts when coding

- i++ increments i by 1 after an operation (post-increment)
  - So you can use i++ instead of i=i+1
- i-- decreases i by 1 after an operation (post-increment)
  - So you can use i-- instead of i=i-1
- ++i increments i by 1 before an operation (pre-increment)
- --i decreases i by 1 before an operation (pre-increment)

What is the difference between pre increment and post increment?

- Try this:

```c
#include <stdio.h>
#include <stdlib.h>
void main()
{
    int i = 1, j = 1, k = 0, m = 0;
    printf("i=%d, j=%d, k=%d, m=%d\n", i, j, k, m);
    k = i++;
    m = ++j;
    printf("i=%d, j=%d, k=%d, m=%d\n", i, j, k, m);

}
```

This means k=i, then increment i

This means first increment j, then m=j

```
i=1, j=1, k=0, m=0
i=2, j=2, k=1, m=2
Press any key to continue . . .
```

There are some other assignment operators:

= += -= *= /= %=

- x += 2 is equivalent to x = x + 2
- x -= 3 is equivalent to x = x – 3
- x *= -5 is equivalent to x = x * -5
- x /= 7 is equivalent to x = x / 7
- x %= 4 is equivalent to x = x % 4

See your textbook for more information on the rules of precedence

- Pretty much the same as MATLAB
- Pre and post increments are included in the rules and can make a difference in results

In C, we can use if statements, switch-case, and loops, but there will be some differences in syntax

<	less than	&&	AND
<=	less than or equal to	||	OR
>	greater than	!	NOT
>=	greater than or equal to		
==	equal to		
!=	not equal to		

Differs from MATLAB

Remember the difference between = and ==

- Tip: Avoid using == with floats or doubles
  - C will check so many significant figures that a very small difference (.0000001) might be interpreted as not equal – depending on your program, this might not be the case

- Instead use something like this:

```
if fabs(a-b)<.0000001
```

One-conditional if statements in C are similar to MATLAB, with some differences in syntax

```
prior commands
if (expression 1)
    {
    commands;
    }
other commands
```

Expression inside (), no semi-colon at the end

Commands within { }

If expression 1 is not true, these commands are skipped

Remember to indent!

No “end” is needed!

If-else statements in C are similar to MATLAB, with some differences in syntax

```
prior commands
if (expression 1)
        {
        commands;
        }
else
        {
        commands;
        }
other commands
```

Commands within { }

No semi-colon at the end

No “end” is needed!

Remember to indent!

If-else-if statements in C are similar to MATLAB, with some differences in syntax

Two separate words!

```
prior commands
if (expression 1)
        {
        commands;
        }
else if (expression 2)
        {
        commands;
        }
else
        {
        commands;
        }
other commands
```

Commands within { }

Remember to indent!

No semi-colon at the end of if, else if, else

No “end” is needed!

Let’s try an example

- Write a program that asks the user to type in number 1, 2, or 3

- If the user types 1, print “You chose 1”; if the user types 2, print “You chose 2”; if the user types 3, print “You chose 3”; otherwise print “You didn’t choose any of the options”

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

int main()
{
    //Create variable
    int your_choice;

    //Request user input
    printf("Please type 1, 2, or 3\n\n");
    scanf("%d", &your_choice);

    //Print appropriate result
    if (your_choice == 1)
    {
        printf("You chose 1\n");
    }
    else if (your_choice == 2)
    {
        printf("You chose 2\n");
    }
    else if (your_choice == 3)
    {
        printf("You chose 3\n");
    }
    else
    {
        printf("You didn't choose any of the options\n");
    }


}
```

Switch-case statements in C are similar to MATLAB, with some differences in syntax

```
prior commands
switch (variable)
{
        case 1:
                command(s);
                break;
        case 2:
                command(s);
                break;
        case 3:
                command(s);
                break;
        default:
                command(s);
}
other commands
```

Must be previously defined; place in ()

Colons at the end

Place everything inside { }

Important!! Break at the end of each case

End with "default", not "otherwise"

Let’s try an example

- Write a program that asks the user to type in number 1, 2, or 3
- If the user types 1, print “You chose 1”; if the user types 2, print “You chose 2”; if the user types 3, print “You chose 3”; otherwise print “You didn’t choose any of the options”
- Now convert to switch-case!

Loops in C are similar to MATLAB, with some differences in syntax

- while loops
- do while loops *(didn't see this in MATLAB)*
- for loops

With while loops, we typically don't know the number of times it will be executed, and it needs to be initialized

Sample typical code:

```
prior commands;
i=5;
        while (expression1)
        {
            commands;
            i=i+1;
        }
other commands;
```

Expression in ();
no semi-colons

No "end"!

Sample alternative code:

```
prior commands;
j=5;
        while (j)
        {
          printf("\nj= %d\n\n",j);
          j=j-1;
        }
```

Evaluated as true/false; false when j=0

Remember to define the variable types for conditions and counters (such as i and j here)

With do-while loops, we typically don't know number of times it will be executed, but will be done at least once

- Doesn't need to be initialized
- Expression is checked after the commands

No semi-colon

```
do
{
        commands;
} while(expression);
```

Semi-colon here

Let’s look at some examples:

1. Write a program that uses a while loop to loop 5 times and print out the loop number

Let’s look at some examples:

1. Write a program that uses a while loop to loop 5 times and print out the loop number

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

int main(void)
{
    int i;
    i = 1;

    //While loop
    while (i <= 5)
    {
        printf("Loop number %d in the while_loop\n", i);
        i = i + 1;
    }


}
```

Let’s look at some examples:

2. Write a program that asks the user to provide a number and then multiplies that number by 9. The program should then ask the user if he/she wants to run the program again. Use a do-while loop.

Let's look at some examples:

2. Write a program that asks the user to provide a number and then multiplies that number by 9. The program should then ask the user if he/she wants to run the program again. Use a do-while loop.

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

int main(void)
{
    //Declare variables
    int k, mult_answer;
    char answer;

    //Use a do-while loop
    do
    {
        //Get input from user
        printf("\nEnter an integer to multiply by 9:\n\n");
        scanf("%d", &k);

        //Multiply
        mult_answer = k * 9;

        //Print to screen
        printf("\nThe answer is: %d\n\n", mult_answer);

        //Ask user - run again?
        printf("\nWould you like to do this again (y/n)?\n\n");
        scanf(" %c", &answer);
    } while (answer == 'y' || answer == 'Y');


}
```

You can use the toupper command to help prevent input errors – it converts a character to an upper case (you need #include <ctype.h>)

```
do
        {
                commands;

                while(toupper(answer)!='Y' && toupper(answer)!='N')
                {
                        printf("\n Error! Type Y or N, please:\n\n");
                        scanf(" %c",&answer);
                }
        } while(toupper(answer)=='Y');
```

With for loops, we execute a loop for a certain number of times, and there are some differences in syntax:

3 sections separated by ;
and enclosed in ()

No ; at the end

```
for (initialization statement; relational expression; increment expression)
{
        commands;
}
```

Let’s look at an example:

- Write a program that uses a for loop to loop 5 times and print out the loop number

Let’s look at an example:

- Write a program that uses a for loop to loop 5 times and print out the loop number

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

void main(void)
{
    int count;

    //For loop
    for (count = 1; count <= 5; count++)
    {
        printf("The counter = %d\n", count);
    }


    printf("\n");

}
```

Here are some variations of the for loop expression:

for (i = start; i < end; i = i + 1)
i, start, end must be declared

for (test = 10; test >= 1; test = test –1)
loop variable can start high to low

for (k = 2; k < 100; k = k * 2)
loop variable can be incremented in any way

for (j = 1; j <= finish; j++)
increment expression can be written in short form:
j++ is same as j = j + 1
j-- is same as j = j – 1
j+=2 is same as j = j + 2
j*=2 is same as j = j * 2

When using loops to access array elements, remember that the first array has index 0, not 1!

- So your control variable must start at 0

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

int main(void)
{
    //Assign values to array a
    int i, a[5] = { 12, 24, 36, 48, 60 };

    //For loop to print elements
    for (i = 0; i<5; i = i + 1)
    {
        printf("Array element[%d] equals %d\n\n", i, a[i]);
    }


}
```

```
Array element[0] equals 12

Array element[1] equals 24

Array element[2] equals 36

Array element[3] equals 48

Array element[4] equals 60

Press any key to continue . . .
```

Note: You will need a nested for loop for a 2-dimensional array

Is your program not working well? Check these:

Troubleshooting slide

- Placement of semi-colons
- Appropriate libraries
- Consistent data types
- All variables declared
- Format for scanf for a char (remember the blank space or use fflush)
- Properties for scanf – remember the &

Note: See troubleshooting slide

Practice Problem

- Write a code to sum the first ***n*** digits (1+2+3+…+n)
- Ask the user to provide the value for n
- Calculate the sum for the n that was provided (use for loop)
- Print the final sum

Submit .cpp file called
"Mena_Time_CForloopTeam#"
("Mena_10am_CForloopL01") into Classwork folder

Note: See troubleshooting slide

Practice Problem

- Write a program asking the user to enter elements of a 2x2 matrix A
- Then display the elements (2 decimal places) as shown in the output
- Use for loops to get the inputs and then to print the values

```
Please enter the elements of Matrix a
Enter element in ROW=1, COL=1: 1
Enter element in ROW=1, COL=2: 2
Enter element in ROW=2, COL=1: 3
Enter element in ROW=2, COL=2: 4
-------------------------------------
You have entered the following elements for Matrix a:
1.00
2.00
3.00
4.00
Press any key to continue . . .
```

Submit .cpp file called “Mena_Time_CMoreLoopsTeam#” (“Mena_10am_CMoreLoopsL01”) into Classwork folder