ENGR 0012 – Engineering Problem Solving

Goals for this week:

- Getting started with C++
- Introduce different data types
- Use input and output commands
- Create arrays

We will be using Geany

- It's free!
- Same for Mac and PC
- Go to Dr. Mandala's page for installation instructions: http://www.pitt.edu/~mam447/
- (Note that your textbook uses Visual Studio; the programming concepts and code remain the same)

In MATLAB, we could run a program one line at a time, but in C we run a complete program

But it's a good idea to run short sections of a program to make sure they are working!

- No prompt (>>) in C

- Good idea to plan out your code before you get started!

Let’s try an example

- Remember to save your file as a .cpp

Libraries need to be included

InClass.cpp

```
1  /*An example program*/
2  #include <stdio.h>
3  #include <stdlib.h>
4  int main()
5  {//this starts the main output/
6      printf("An example program");
7      printf("to learn C");
8  }
```

Libraries

We need to let C know which libraries we need (libraries have the built-in functions and commands). C will only search within the libraries we include (unlike MATLAB)!

Examples of the commands in the <stdio.h> library are:

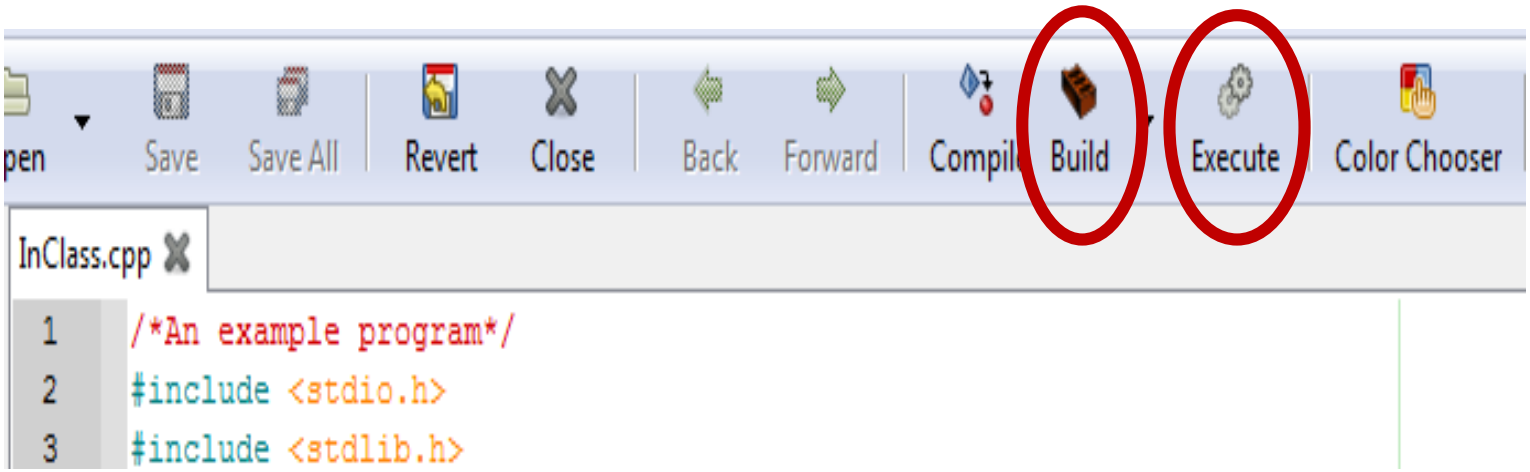
- printf → prints output enclosed in “ ” to the screen
- fprintf → prints output to a file
- scanf → reads input from the screen
- fscanf → reads input from a file
- fopen → opens a file for reading and writing
- fclose → closes a file

Examples of other libraries in C are:

- <stdio.h> → standard input/output
- <ctype.h> → character classification and character conversion
- <math.h> → math functions
- <string.h> → string handling

- (See your textbook!)

To run the code in Geany, you need to Build, then Execute

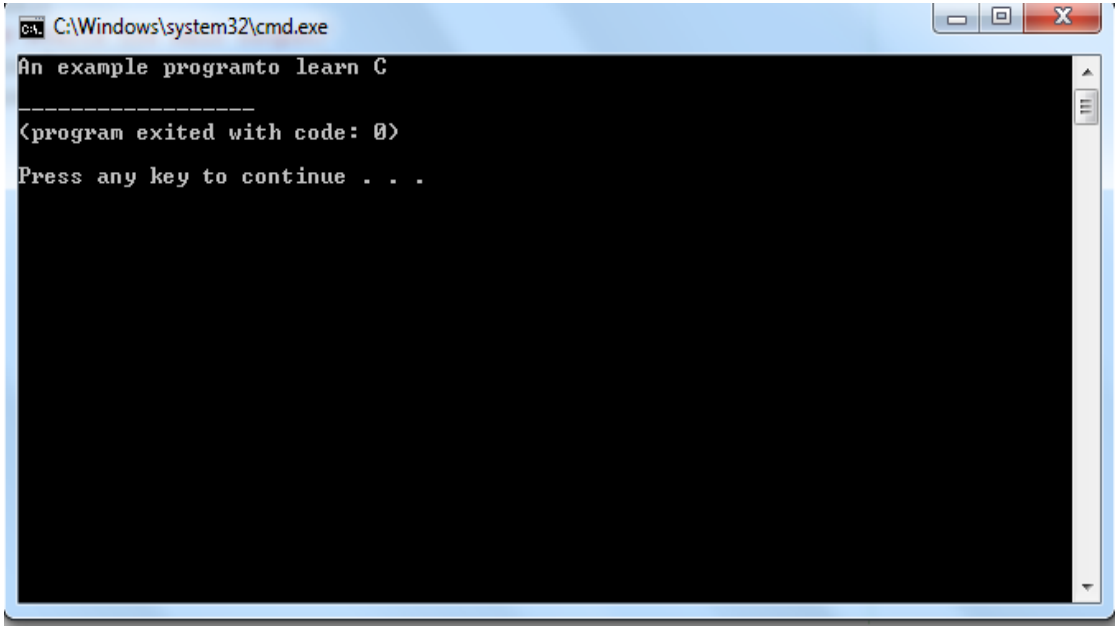


Red squiggly lines will point out errors, and details about these errors will be provided

Status
Compiler
Messages
Scribble

```
 void main(void)
                ^
InClass.cpp: In function 'int main()':
InClass.cpp:8:3: error: expected ';' before 'system'
   system("pause");
   ^
Compilation failed.
```

The output:



Let’s take a better look at our code:

Use /* */ to start and end comments

InClass.cpp

```
1  /*An example program*/
2  #include <stdio.h>
3  #include <stdlib.h>
4  int main()
5  {//this starts the main output/
6      printf("An example program");
7      printf("to learn C");
8  }
```

Preprocessor directive, to link library files

Use // when the rest of the line is a comment

Let’s take a better look at our code:

End commands with ;

InClass.cpp

```
1  /*An example program*/
2  #include <stdio.h>
3  #include <stdlib.h>
4  int main()
5  {//this starts the main output/
6      printf("An example program");
7      printf("to learn C");
8  }
```

Every program has one function main, enclosed in { }

Sends output to screen

Notice that we have two lines with the printf command, but our output is all in one line

InClass.cpp

```
1  /*An example program*/
2  #include <stdio.h>
3  #include <stdlib.h>
4  int main()
5  {//this starts the main output/
6      printf("An example program");
7      printf("to learn C");
8  }
```

C:\Windows\system32\cmd.exe

```
An example programto learn C
------------------
(program exited with code: 0)

Press any key to continue . . .
```

In C, you have to insert a line break whenever you need one: use \n

InClass.cpp

```
1  /*An example program*/
2  #include <stdio.h>
3  #include <stdlib.h>
4  int main()
5  {//this starts the main output/
6      printf("An example program\n");
7      printf("to learn C\n");
8  }
```

C:\Windows\system32\cmd.exe

```
An example program
to learn C


------------------
(program exited with code: 0)

Press any key to continue . . .
```

If you want to display characters that are also used as commands, use an escape sequence (see your textbook)

Escape sequence	Meaning
\’	Displays a single quote
\”	Displays a double quote
\\n	Displays the characters \n
\\t	Displays the characters \t
%%	Displays a percent symbol %

Try this! Can you get C to print this:

```
The symbol for quotation marks is "
The symbol for percent is %
I have to remember the \n to insert a new line
Press any key to continue . . .
```

From course textbook

In C, we have to define the different variable types

- Is it an integer, a decimal, …?
- Within our program, that variable will remain as it was defined – once it was defined, it can't be changed!

Why? Because of the way computers store data: using bits (a single character, 0 or 1) and bytes (eight bits) – see your textbook!

While there are many different data types we can use in C, we will mostly use four:

- int → integer quantities, with no commas, no decimal point
- float → quantities that may include a decimal component and/or an exponent
- double → similar to float, but more significant figures
- char → a single character (strings will be character arrays)

When naming variables, follow the conventions we used in MATLAB:

YES:

- Use UPPER and lower case letters, numbers and underscores – names are case sensitive!
- Variable names must begin with a letter
- Variable names can be up to and including 31 characters
- Use descriptive names!

NO:

- Space between characters in a variable name
- Names that differ by only one letter
- Meaningless names
- Same name for two different variables
- Names that are already used in C libraries

When defining a variable, do so within the main (usually at the top, beneath the opening bracket)

For example:

When defining a variable, do so within the main (at the top, beneath the opening bracket)

Use int to declare an integer (no decimal points)

Use double similarly to float, but when more significant figures are needed

```
/* Declaring variables*/

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

int main()
{
    int a, b;
    float x, y;
    double u, v;
    char c;
    char name[20];

}
```

Use float to declare values with decimals and/or exponents

Use char to declare a single character or an array of characters

To assign a value to a variable, use =, scanf, fscanf, or #define

- Use = when you declare the data type

```
float x=5.0;
```

- Or use = anytime later in the program

```
y=2*x;
```

To assign a value to a variable, use =, scanf, fscanf, or #define

- Use scanf after you declared the data type

- This is similar to input in MATLAB, but you don't have the option of including text for the user

If you want to include text for the user, include a printf command prior to the scanf!

To assign a value to a variable, use =, scanf, fscanf, or #define

- Use this format:

  scanf("format_string", variables with"&" preceding the variable name separated by commas);

- For example

  scanf("%lf %lf", &u, &v);

Data type, beginning with %

Variable name, beginning with &

To represent each data type: %d → int
%f → float
%lf → double
%c → char
%s → strings

Let’s try an example

Let’s try an example

```c
/* Declaring variables*/

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

int main()
{
    int a, b;
    float x, y;
    double u, v;
    char c;
    char name[20];

    printf("What is the value of x?");
    scanf("%f", &x);

    printf("The value of x is %f\n", x);


}
```

- Note: be careful when using scanf with char!

- Use:

```
scanf(" %c", &VarName)
```

Space

- Or use:

```
fflush(stdin);
scanf("%c", &VarName)
```

No space

To assign a value to a variable, use =, scanf, fscanf, or #define

- Use #define before the main
- Use to define a variable that will be used throughout the program
- DO NOT change the value of the variable within the program

To assign a value to a variable, use =, scanf, fscanf, or #define

```
/* Declaring variables using define*/

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

#define Var2 53.4

void main()
{
    printf("%f", Var2);
    printf("\n\nVar2 = %f\n\n", Var2);

}
```

Write variable name, followed by variable value, without commas or semi-colons

We can use printf to display information, and we've already used it in two different ways:

```
printf("An example program");

printf("\n\nVar2 = %f\n\n", Var2);
```

Similar to scanf, but we don't need & in front of the variable name

Practice Problem

Write a program that:

1. Declares four double variables: x, y, u, v
2. Declares one char variable (Letter)
3. Assigns x=5
4. Prints “I like C” to the screen
5. Assigns a value to y (y=2x)
6. Prints the values of x and y
7. Requests values of u and v from user
8. Prints values of u and v
9. Requests user’s favorite letter
10. Prints “You entered the letter *Letter*”

You can customize your output in C, by remembering this format:

%[flag][field width][.precision]format type

Field width lets you specify how many characters your output will have

Try this:

```c
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
    int a = 5;
    printf("\na = %d, a = % 5d\n\n", a, a);

}
```

```
a = 5, a =     5

Press any key to continue . . .
```

You can customize your output in C, by remembering this format:

%[flag][field width][.precision]format type

Precision lets you specify how many decimal points to include

Try this:

```c
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
    float a = 3.2;
    printf("\na = %f, a = %5.2f\n\n", a, a);

}
```

```
a = 3.200000, a =  3.20

Press any key to continue . . .
```

You can customize your output in C, by remembering this format:

%[flag][field width][.precision]format type

Flag lets you justify your output and include zeros:
- Nothing: default is to right justify the output
- “-”: output will be left justified in the field width
- “+”: output will be right justified in the field width and a “+” will be displayed if the value is positive
- “0”: leading zeroes will be added so the output uses the field width

Try this:

```c
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
    float a = 3.2;
    printf("\na = %f, a = %-9.2f\n\n", a, a);
    printf("a = %+9.2f, a = %09.2f\n\n", a, a);

}
```

```
a = 3.200000, a = 3.20

a =      +3.20, a = 000003.20

Press any key to continue . . .
```

When using C, be careful!

- Syntax errors are not the only type of error you will encounter

Can you find the error here?

```c
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
    float a = 3.2;
    printf("\na = %-9.2d\n\n", a);

}
```

When using C, be careful!

- Syntax errors are not the only type of error you will encounter

Can you find the error here?

```c
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
    float a = 3.2;
    printf("\na = %-9.2d\n\n", a);

}
```

C will give you an output, but it will be wrong!

```
a = -1610612736

Press any key to continue . . .
```

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);


}
```

(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);


}
```

Practice Problem

```
Please enter a number for bananas:
2
Please enter a number for oranges:
3
Please enter a number for grapes:
4
Please enter the first letter of your first name:
i

apples = 5, bananas = 2

oranges = 3.000000, grapes = 4.000000

First Letter = i

Pi = 3.141560

This was fun!

Press any key to continue . . .
```

Write a program that will do the following:

1. Declare 2 integer variables: apples, bananas
2. Set apples equal to 5
3. Declare 2 double variables: oranges, grapes
4. Declare one character variable: FirstLetter
5. Use #define to create a variable PI equal to 3.14156
6. Ask user to provide values for bananas, oranges, and grapes
7. Ask user to provide the first letter of his/her first name
8. Display the variables and text as shown in the output

Submit .cpp file called "Mena_Time_CIntroGr#" ("Mena_10am_CIntroL01") into Classwork folder

C can work with one- to n- dimensional arrays, but notice how C differs from MATLAB:

MATLAB

- Use () and commas for indices → A(2,5)
- Starts index numbers at 1
- Length or data type don't matter
- Show array elements with [] → B=[1 2 3]

C

- Use separate [] for indices → A[2][5]
- Starts index numbers at 0
- Define length and size before using
- Show array elements with {} and commas → B={1, 2, 3}

There are many ways to declare one-dimensional arrays in C – try this:

```
The first element in array a is: 4

The second element in array b is: 2

The third element in array c is: 7

The second element in array d is: 2.000000
Press any key to continue . . .
```

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

#define N 3

int main()
{
    //Declare arrays and assign values
    int a[2] = { 4, 3 };
    int b[3] = { 1, 2 }; //Didn't provide all elements
    int c[] = { 5, 6, 7 }; //Didn't specify dimension

    double d[N];

    //Assign values element by element
    d[0] = 1;
    d[1] = 2;
    d[2] = 3;

    //Print values to screen
    printf("The first element in array a is: %d", a[0]);
    printf("\n\nThe second element in array b is: %d", b[1]);
    printf("\n\nThe third element in array c is: %d", c[2]);
    printf("\n\nThe second element in array d is: %lf", d[1]);
    printf("\n");

}
```

There are many ways to declare two-dimensional arrays in C:

```
int a[2][3] = {5, -3, 0, 1, 2, 3};
int a[][3] = {5, -3, 0, 1, 2, 3};
int a[2][3] = {{5, -3, 0},{1,2,3}}
int a[2][3] = {{5, -3, 0},
               {1,2,3}}
```

All work for
matrix a= 5 -3 0
1 2 3

In C, order of storage for 2-d arrays is by rows – see your textbook!

For example:

```
C:\Users\imena\documents\visual studio 2015\
Element (1,1) in matrix a is: 5

Element (2,2) in matrix b is: 2

Element (2,3) in matrix c is: 3

Element (1,3) in matrix d is: 0
Press any key to continue . . .
```

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

int main()
{
    //Declare 2-d arrays and assign values
    int a[2][3] = { 5, -3, 0, 1, 2, 3 };
    int b[][3] = { 5, -3, 0, 1, 2, 3 };
    int c[2][3] = { { 5, -3, 0 },{ 1,2,3 } };
    int d[2][3] = { { 5, -3, 0 },{ 1,2,3 } };

    //Print values to screen
    printf("Element (1,1) in matrix a is: %d", a[0][0]);
    printf("\n\nElement (2,2) in matrix b is: %d", b[1][1]);
    printf("\n\nElement (2,3) in matrix c is: %d", c[1][2]);
    printf("\n\nElement (1,3) in matrix d is: %d", d[0][2]);
    printf("\n");

}
```

Using same matrix =

$$\begin{matrix} 5 & -3 & 0 \\ 1 & 2 & 3 \end{matrix}$$

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");

}
```

C:\Users\imena\documents\visual studio 2015\Projects\Project1\Deb

```
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 . . .
```

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?

```
C:\Users\imena\documents\visual studio 2015\Projects
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 CoruseWeb

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