C++ Crash Course 1

Due May 18 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** cpp

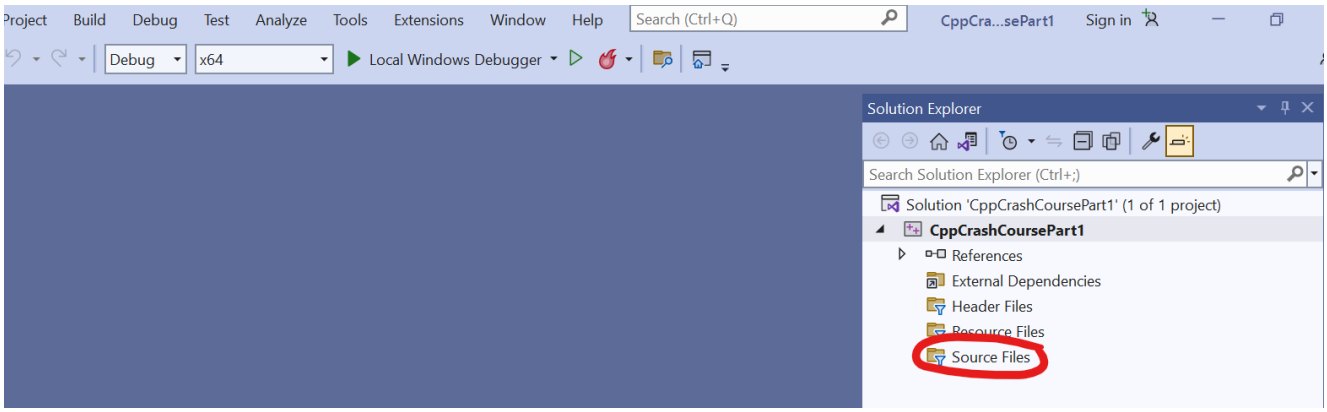
The objective of this assignment is to gain a basic understanding of how to write C++ code to solve engineering problems. You will learn the following principles of C++ programming:

1. How to create, compile, and run a C++ program in Visual Studio
2. How and why to include libraries in your C++ programs
3. How to declare variables and what data types are
4. How to write outputs to the console window
5. How to perform common engineering scalar calculations
6. How to write functions that can return one variable
7. How to use pointers to enable functions to return multiple variables
8. How to create basic arrays and matrices
9. How to perform basic calculations with arrays and matrices

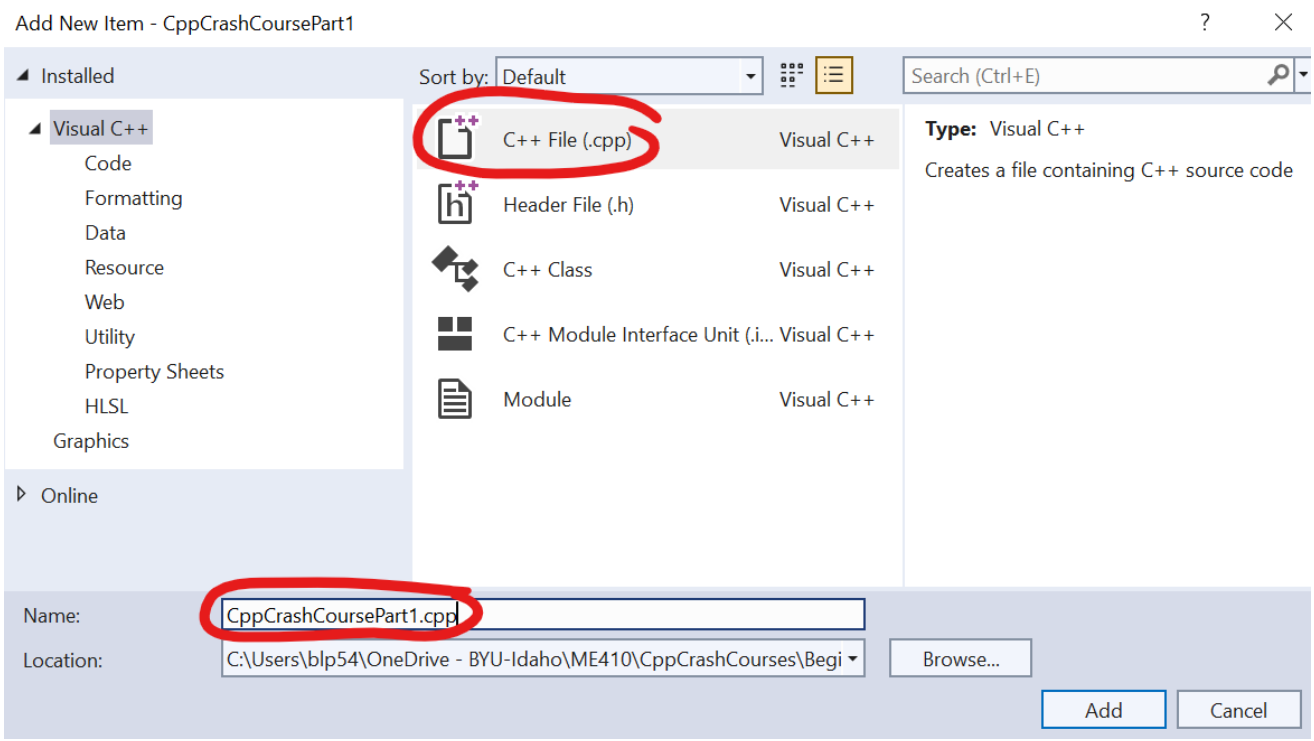
Do the following:

Hello World!

- Open Visual Studio
- Select *Create a new project*
- Select *Empty Project* and click *Next*
- Click the three dots ... next to the folder location, and navigate to a folder location where you would like to save the project
- Name the project *CppCrashCoursePart1_yourname*
- Check the box next to *Place solution and project in the same directory*
- Select *Create*



- Right click on *Source Files* in the *Solution Explorer* window and select *Add >> New Item*

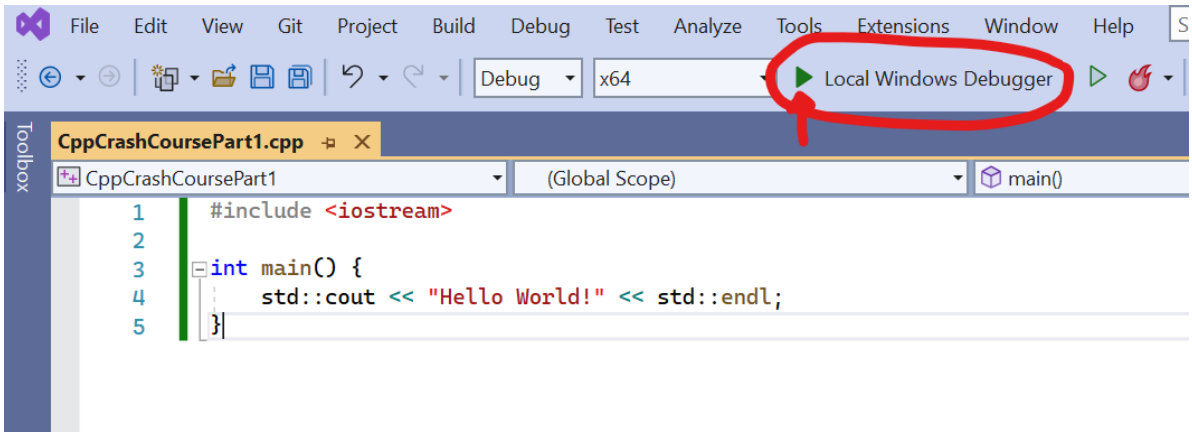


- In the *Add New Item* window, select *C++ File (.cpp)*, change the *Name* to *CppCrashCoursePart1.cpp* and click *Add*
- A new window should open with the *CppCrashCoursePart1.cpp* file that you just added. (If not, you can open it by double-clicking *CppCrashCoursePart1.cpp* under *Source Files* in the *Solution Explorer*)
- **Instead of copying the code in this assignment, it is recommended that you type it. Typing it helps you learn better, and may prevent compiler errors that are caused by incompatible font formats between Canvas and Visual Studio.**
- Type the following code in the *CppCrashCoursePart1.cpp* file:

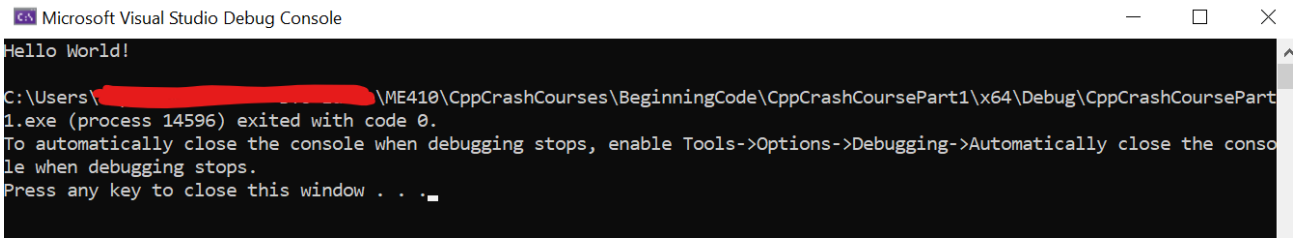
```
#include <iostream> //Add this command to use std::cout and std::endl

int main() { //The MAIN function is the required entry point for a C++ program
    std::cout << "Hello World!" << std::endl; //COUT prints "Hello World!" to the console.  ENDL moves the cursor to a new line.
    return 0; //The MAIN function should return an integer
}
```

Then click the green play button next to *Local Windows Debugger*.



It should compile the code and produce a window that says *Hello World!* It should look something like



Congratulations! You have written and compiled a C++ program.

Explanation of the Hello World Code

The *Hello World* code began with a command to include the *iostream* library:

```
#include <iostream>
```

This library allows the C++ program to use the commands *cout* and *endl*. The *cout* command prints letters, numbers, and other characters to the console window. The *endl* command is a carriage return that causes the cursor in the console window to move to the next line. A comment in C++ is made using two forward slashes //. Whatever follows the two forward slashes is not part of the compiled code. For example, the phrase

```
//Add this command to use std::cout and std::endl
```

that follows the *#include <iostream>* command is ignored by the C++ compiler. It is not part of the compiled program. Comments are helpful to let programmers write explanations about the code. The next line is the *main()* function:

```
int main() {}
```

Each C++ program needs an entry point, or code that is called first. For console programs like the one we created, the *main()* function is a required entry point. It is the function that C++ calls first. (Later we will create other functions, and they will be called from within the *main()* function). The code that is within the curly braces {} is part of the *main()* function. It is said to be within the *scope* of the *main()* function. The *main()* function returns an *int* data type. The *int* data type is an integer which is a whole number. Our program returned zero using the command

```
return 0;
```

The part of the program that printed *Hello World* to the console window was the line

```
std::cout << "Hello World!" << std::endl;
```

The *cout* command prints letters, numbers, and other characters to the console window. The *endl* command is a carriage return that causes the cursor in the console window to move to the next line. The *cout* command is part of the *std* namespace, meaning that *cout* will not work without using the identifier *std::* or by using the command *using namespace std*. Understanding namespace is a more advanced topic that can be explored later. For now, just know that to use *cout* and *endl*, they must be used with the *std* identifier.

Common Calculations for Engineers

Engineers use math. To use many of the basic math operators, C++ code will need to include the library *math.h*. The *math.h* library includes at least the following math operations:

cos, sin, tan, acos, asin, atan, atan2, cosh, sinh, tanh, acosh, asinh, atanh, exp, frexp, ldexp, log, log10, modf, exp2, expm1, ilogb, log1p, log2, logb, scalbn, scalbln, pow, sqrt, cbrt, hypot, erf, erfc, tgamma, lgamma, ceil, floor, fmod, trunc, round, lround, llround, rint, lrint, llrint, nearbyint, remainder, remquo, copysign, nan, nextafter, nexttoward, fdim, fmax, fmin, fabs, abs, fma
To learn more about how to use these operators, do an Internet search using the keyword *math.h*. Modify your *CppCrashCoursePart1.cpp* file as follows:

```
#include <iostream> //Add this command to use std::cout and std::endl
#include <math.h> //For math operations: sqrt, pow, abs, sin, cos, tan, asin, acos, atan, atan2

using namespace std; //allows us to remove std:: from std::cout and std::endl

int main() { //The MAIN function is the required entry point for a C++ program
    cout << "Hello World!" << endl; //COUT prints "Hello World!" to the console.  ENDL moves the cursor to a new line.

    //Perform very basic algebra such as add, subtract, multiply, and divide.
    //This does not require math.h
    double sum = 2.0 + 3.0;
    double difference = 2.0 - 3.0;
    double product = 2.0 * 3.0;
    double quotient = 2.0 / 3.0;
    cout << "sum = " << sum << endl; //Print the sum to the console window
    cout << "difference = " << difference << endl; //Print the difference to the console window
    cout << "product = " << product << endl; //Print the product to the console window
    cout << "quotient = " << quotient << endl; //Print the quotient to the console window

    //Perform math that does require the math.h library
    const double pi = 3.14159265358979323846264338327 9;
    double sin2pi = sin(2.0 * pi);
    double atan2_4_n2 = atan2(4.0, -2.0);
    double sixCubed = pow(6.0, 3.0);
    double sqrt_5 = sqrt(5.0);
    cout << "sin(2.0 * pi) = " << sin2pi << endl; //Print the sin of 2*pi to the console window
    cout << "atan2(4.0, -2.0) = " << atan2_4_n2 << endl; //Print the arctangent of 4/(-2) to the console window
    cout << "pow(6.0, 3.0) = " << sixCubed << endl; //Print 6^3 to the console window
    cout << "sqrt(5.0) = " << sqrt_5 << endl; //Print the square root of 5 to the console window

    return 0; //The MAIN function should return an integer
}
```

Click the green play button next to *Local Windows Debugger*. Did the math operators work correctly? You may want to compare the result with what you get in MATLAB or some other program.

Some keywords in C++

You may have noticed some new keywords in the C++ code above. May of these keywords tell the C++ compiler what type of data it needs to create memory for. The data type *double* in the phrase *double sum = 2.0 + 3.0;* indicates that *sum* is a floating point number with double precision. Floating point numbers have decimal places, unlike integers. You may have also noticed the sum used the numbers *2.0* and *3.0* instead of 2 and 3. The C++ compiler automatically treats the numbers 2 and 3 like integers, but it treats *2.0* and *3.0* as double precision numbers. When writing code in C++, it can be extremely important to include the decimal on floating point numbers. Forgetting the decimal can lead to unexpected results than are difficult to debug.

The phrase *const double pi = 3.14159265358979323846264338327 9;* defined *pi* as a constant that cannot be changed later in the C++ code. If an attempt was made to change *pi* to a different number later in the program, the compiler would throw an error.

The phrase *using namespace std* allows the C++ program to use the *cout* and *endl* commands directly without the *std::* identifier.

Functions that Return One Variable

Functions in C++ require a function declaration, a function definition, and a call to the function. The declaration tells the compiler that the function exists. The definition contains the actual body of the function. The call to the function executes it. The function definition can double-count as the declaration if it is placed in the C++ code before it is called.

Modify your code to include the *sind()* function as shown below. The *sind()* function calculates the sine of an angle in degrees. The angle in degrees has a double precision data type, and the function returns a double precision number.

```
#include <iostream> //Add this command to use std::cout and std::endl
#include <math.h> //For math operations: sqrt, pow, abs, sin, cos, tan, asin, acos, atan, atan2

using namespace std; //allows us to remove std:: from std::cout and std::endl
const double pi = 3.14159265358979323846264338327 9;

//function declaration of sind()
double sind(const double angle_degrees);

int main() { //The MAIN function is the required entry point for a C++ program
    cout << "Hello World!" << endl; //COUT prints "Hello World!" to the console.  ENDL moves the cursor to a new line.
```

```
    //Perform very basic algebra such as add, subtract, multiply, and divide.
    //This does not require math.h
    double sum = 2.0 + 3.0;
    double difference = 2.0 - 3.0;
    double product = 2.0 * 3.0;
    double quotient = 2.0 / 3.0;
    cout << "sum = " << sum << endl; //Print the sum to the console window
    cout << "difference = " << difference << endl; //Print the difference to the console window
    cout << "product = " << product << endl; //Print the product to the console window
    cout << "quotient = " << quotient << endl; //Print the quotient to the console window

    //Perform math that does require the math.h library
    double sin2pi = sin(2.0 * pi);
    double atan2_4_n2 = atan2(4.0, -2.0);
    double sixCubed = pow(6.0, 3.0);
    double sqrt_5 = sqrt(5.0);
    cout << "sin(2.0 * pi) = " << sin2pi << endl; //Print the sin of 2*pi to the console window
    cout << "atan2(4.0, -2.0) = " << atan2_4_n2 << endl; //Print the arctangent of 4/(-2) to the console window
    cout << "pow(6.0, 3.0) = " << sixCubed << endl; //Print 6^3 to the console window
    cout << "sqrt(5.0) = " << sqrt_5 << endl; //Print the square root of 5 to the console window

    //Function call to sind
    double sin_45 = sind(45.0);
    cout << "sin(45 degrees) = " << sin_45 << endl;

    return 0; //The MAIN function should return an integer
}

//function definition of sind()
double sind(const double angle_degrees) {
    double angle_radians = angle_degrees * pi / 180.0;
    return (sin(angle_radians));
}
```

Pay special attention to the following details. First, *const double pi = 3.14159265358979323846264338327 9;* was moved outside the scope of the *main()* function. This was necessary so that it could be used in both the *main()* and *sind()* functions. Before, *pi* was only defined within the curly braces {} of the *main()* function. Second, notice that the function declaration came before the function call. That allowed the code to place the function definition after the function call. Finally, notice that the *sind()* function returned the double-precision value calculated by *sin(angle_radians)*.

Do It Yourself

Create your own function named *cosd()*. It should take an input angle in degrees and output the cosine of the angle. Print the result to the C++ console window.

Functions that Return Multiple Variables

If a C++ function must return multiple variables, it must pass them by reference. C++ uses pointers, denoted with asterisks (*), to pass variables by reference. When passing a variable by reference, instead of passing the variable to the function, the function call passes the memory location of the variable to the function using the *&* operator. The function uses a pointer (*) to change the variable in the memory location that was passed to it.

Modify your code as follows to create a function that can return multiple variables.

```
#include <iostream> //Add this command to use std::cout and std::endl
#include <math.h> //For math operations: sqrt, pow, abs, sin, cos, tan, asin, acos, atan, atan2

using namespace std; //allows us to remove std:: from std::cout and std::endl
const double pi = 3.14159265358979323846264338327 9;

//function declaration of sind()
double sind(const double angle_degrees);

//This is both the function declaration and definition.  It must come before the function call
void manyOutputs(
    double* output1, //first output
    double* output2, //second output
    const double input1, //first input
    const double input2, //second input
    const double input3 //third input
) {
```

```
    //Access the variable in the memory location of output1 to change it
    *output1 = input1 + input2;

    //Access the varaible in the memory location of output2 to change it
    *output2 = input2 * input3;
}

int main() { //The MAIN function is the required entry point for a C++ program
    cout << "Hello World!" << endl; //COUT prints "Hello World!" to the console.  ENDL moves the cursor to a new line.

    //Perform very basic algebra such as add, subtract, multiply, and divide.
    //This does not require math.h
    double sum = 2.0 + 3.0;
    double difference = 2.0 - 3.0;
    double product = 2.0 * 3.0;
    double quotient = 2.0 / 3.0;
    cout << "sum = " << sum << endl; //Print the sum to the console window
    cout << "difference = " << difference << endl; //Print the difference to the console window
    cout << "product = " << product << endl; //Print the product to the console window
    cout << "quotient = " << quotient << endl; //Print the quotient to the console window

    //Perform math that does require the math.h library
    double sin2pi = sin(2.0 * pi);
    double atan2_4_n2 = atan2(4.0, -2.0);
    double sixCubed = pow(6.0, 3.0);
    double sqrt_5 = sqrt(5.0);
    cout << "sin(2.0 * pi) = " << sin2pi << endl; //Print the sin of 2*pi to the console window
    cout << "atan2(4.0, -2.0) = " << atan2_4_n2 << endl; //Print the arctangent of 4/(-2) to the console window
    cout << "pow(6.0, 3.0) = " << sixCubed << endl; //Print 6^3 to the console window
    cout << "sqrt(5.0) = " << sqrt_5 << endl; //Print the square root of 5 to the console window

    //Function call to sind
    double sin_45 = sind(45.0);
    cout << "sin(45 degrees) = " << sin_45 << endl;

    //Function call to manyOutputs()
    double out1, out2;
    manyOutputs(&out1, &out2, 1.0, 2.0, 3.0);
    cout << "out1 = " << out1 << " and out2 = " << out2 << endl;

    return 0; //The MAIN function should return an integer
}

//function definition of sind()
double sind(const double angle_degrees) {
    double angle_radians = angle_degrees * pi / 180.0;
    return (sin(angle_radians));
}
```

Pay attention to how the function call to *manyOutputs()* passed the memory locations (*&out1* and *&out2*) of the function outputs to the function *manyOutputs()*. The function referenced the values in the memory locations using the pointer commands **output1 = input1 + input2;* and **output2 = input2 * input3;*. The function *manyOutputs()* never used the *return*() command. That is because it was of data type *void*. Void functions do not return anything directly, and can only return variables by passing them by reference.

Do It Yourself

Create a function named divideAndMultiply(). The function should have two inputs and two outputs. When calling the function, the two inputs should be the numbers 4 and 5. The two outputs should be the quotient (4/5) and product (4*5) of the inputs. Use the concept of passing by reference. Print the results to the console window.

Basic Arrays and Matrices

C++ has many ways of working with arrays and matrices. This tutorial teaches one of the most basic ways. Modify your code as follows:

```
#include <iostream> //Add this command to use std::cout and std::endl
#include <math.h> //For math operations: sqrt, pow, abs, sin, cos, tan, asin, acos, atan, atan2

using namespace std; //allows us to remove std:: from std::cout and std::endl
const double pi = 3.14159265358979323846264338327 9;

//function declaration of sind()
double sind(const double angle_degrees);

//This is both the function declaration and definition.  It must come before the function call
```

```
void manyOutputs(
    double* output1, //first output
    double* output2, //second output
    const double input1, //first input
    const double input2, //second input
    const double input3 //third input
) {
    //Access the variable in the memory location of output1 to change it
    *output1 = input1 + input2;

    //Access the varaible in the memory location of output2 to change it
    *output2 = input2 * input3;
}

//Declare the matrixMultiply function C = A * B
void matrixMultiply(double C[2][4], const double A[2][3], const double B[3][4]);

int main() { //The MAIN function is the required entry point for a C++ program
    cout << "Hello World!" << endl; //COUT prints "Hello World!" to the console.  ENDL moves the cursor to a new line.

    //Perform very basic algebra such as add, subtract, multiply, and divide.
    //This does not require math.h
    double sum = 2.0 + 3.0;
    double difference = 2.0 - 3.0;
    double product = 2.0 * 3.0;
    double quotient = 2.0 / 3.0;
    cout << "sum = " << sum << endl; //Print the sum to the console window
    cout << "difference = " << difference << endl; //Print the difference to the console window
    cout << "product = " << product << endl; //Print the product to the console window
    cout << "quotient = " << quotient << endl; //Print the quotient to the console window

    //Perform math that does require the math.h library
    double sin2pi = sin(2.0 * pi);
    double atan2_4_n2 = atan2(4.0, -2.0);
    double sixCubed = pow(6.0, 3.0);
    double sqrt_5 = sqrt(5.0);
    cout << "sin(2.0 * pi) = " << sin2pi << endl; //Print the sin of 2*pi to the console window
    cout << "atan2(4.0, -2.0) = " << atan2_4_n2 << endl; //Print the arctangent of 4/(-2) to the console window
    cout << "pow(6.0, 3.0) = " << sixCubed << endl; //Print 6^3 to the console window
    cout << "sqrt(5.0) = " << sqrt_5 << endl; //Print the square root of 5 to the console window

    //Function call to sind
    double sin_45 = sind(45.0);
    cout << "sin(45 degrees) = " << sin_45 << endl;

    //Function call to manyOutputs()
    double out1, out2;
    manyOutputs(&out1, &out2, 1.0, 2.0, 3.0);
    cout << "out1 = " << out1 << " and out2 = " << out2 << endl;

    //Define two arrays with three elements each and calculate their cross product
    float vec1[3] = { 1.0f, 2.0f, 3.0f };
    float vec2[3] = { 3.0f, 2.0f, 1.0f };
    float crossProduct[3];
    crossProduct[0] = vec1[1] * vec2[2] - vec2[1] * vec1[2];
    crossProduct[1] = vec1[2] * vec2[0] - vec2[2] * vec1[0];
    crossProduct[2] = vec1[0] * vec2[1] - vec2[0] * vec1[1];
    cout << "cross(vec1, vec2) = [  ";
    for (float& elem : crossProduct) {
        cout << elem << "  ";
    }
    cout << "]" << endl;

    //Multiply a 2x3 matrix by a 3x4 matrix C = A * B
    double A[2][3] = { {1.0,2.0,3.0},{3.0,2.0,1.0} };
    double B[3][4] = { {1.0,2.0,3.0,4.0},{5.0,6.0,7.0,8.0},{9.0,10.0,11.0,12.0} };
    double C[2][4];
    matrixMultiply(C, A, B);
    cout << "C = [ ";
    for (unsigned int ii = 0; ii < 2; ii++) {
        for (int jj = 0; jj < 4; jj++) {
            cout << C[ii][jj] << " ";
        }
        if (ii < 1)
            cout << endl;
        else
            cout << "]" << endl;
    }


    return 0; //The MAIN function should return an integer
}

//function definition of sind()
double sind(const double angle_degrees) {
```

```
    double angle_radians = angle_degrees * pi / 180.0;
    return (sin(angle_radians));
}

//Define the matrixMultiply function C = A * B
void matrixMultiply(double C[2][4], const double A[2][3], const double B[3][4]) {
    for (int ii = 0; ii < 2; ii++) {
        for (int jj = 0; jj < 4; jj++) {
            C[ii][jj] = 0.0;
            for (int kk = 0; kk < 3; kk++) {
                C[ii][jj] += A[ii][kk] * B[kk][jj];
            }
        }
    }
}
```

Pay special attention to how matrices are passed to the function *matrixMultiply()*. This approach requires the matrix sizes to be included in the function declaration and definition. Passing matrices without specifying their sizes is beyond the scope of this assignment but will be taught in later C++ Crash Courses. The function definition of *matrixMultiply()* includes three nested *for* loops. The line *C[ii][jj] += A[ii][kk] * B[kk][jj];* is equivalent to *C[ii][jj] = C[ii][jj] + A[ii][kk] * B[kk][jj];*.

The *for* loop for printing the cross product includes the phrase *for (float& elem : crossProduct)*. This phrase is especially useful when the size of the array (crossProduct in this example) is unknown. It iterates through each element of the crossProduct and assigns the element to the floating point value *elem*. The *float* data type is for single precision floating point (decimal) numbers.

Do It Yourself

Create a function that can add two 2x2 matrices: A = { {1.0, 2.0}, {3.0, 4.0} } and B = { {5.0, 6.0}, {7.0, 8.0} }. Print the result C = A + B; to the console window.

What to Submit

Demonstrate to the instructor that your code can do the following:

- Use a function cosd() to calculate the cosine of an angle given in degrees instead of radians
- Use the function divideAndMultiply(). The function should have two inputs and two outputs. When calling the function, the two inputs should be the numbers 4 and 5. The two outputs should be the quotient (4/5) and product (4*5) of the inputs.
- Use a function to add the two 2x2 matrices: A = { {1.0, 2.0}, {3.0, 4.0} } and B = { {5.0, 6.0}, {7.0, 8.0} }. Print the result C = A + B; to the console window.

Submit your C++ file.