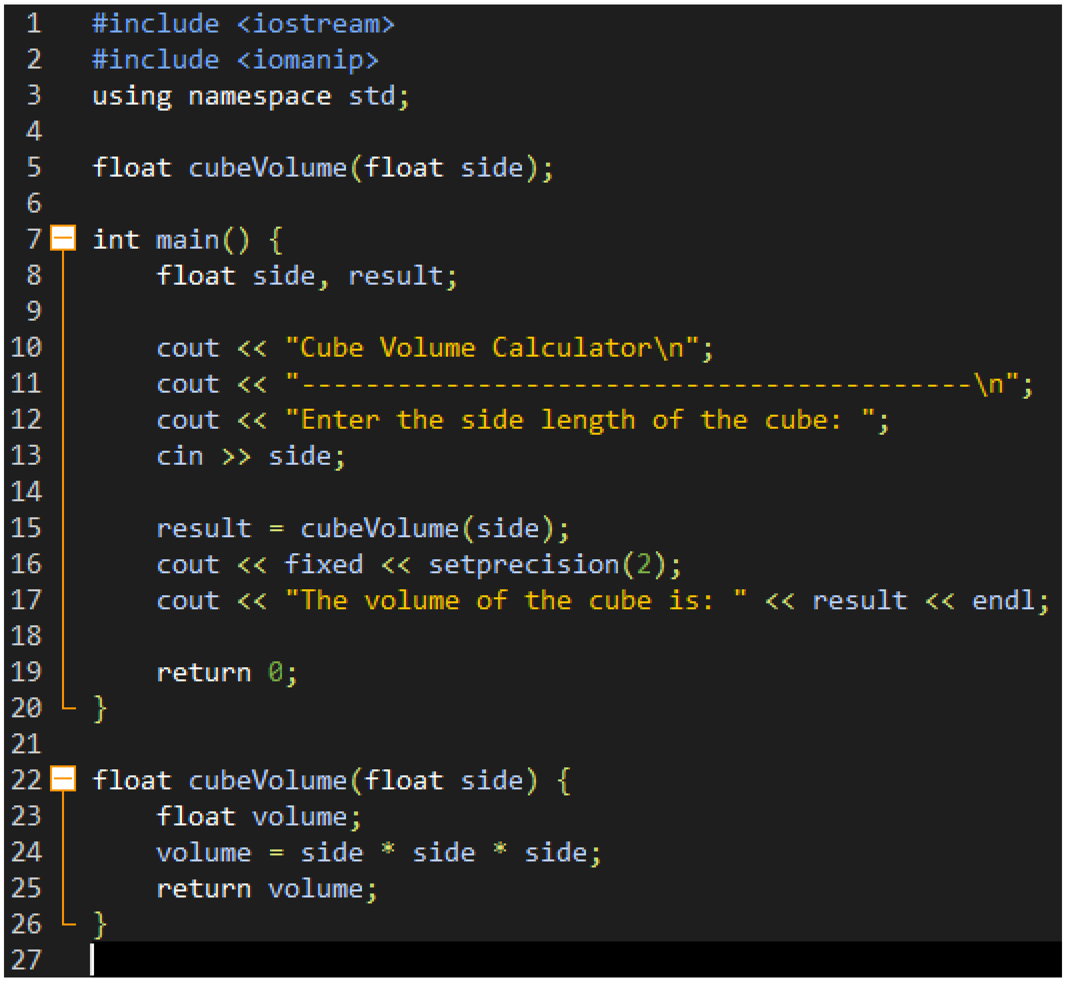
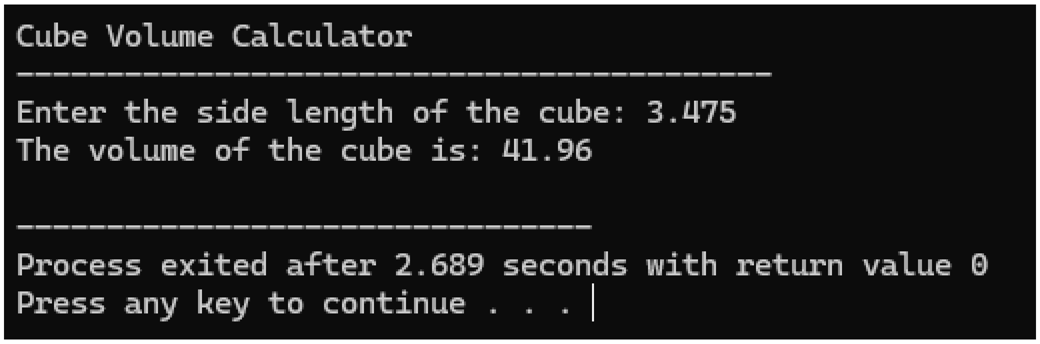
Hands-on Activity 6.2	
Built-in Functions	
Course Code: CPE 007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed: 10/30/2025**
Section: CPE11S1	**Date Submitted: 10/30/2025**
Name(s): Juan Paulo C. Lara	**Instructor: Engr. Jimlord M. Quejado**
6. Output	

#1
CODE



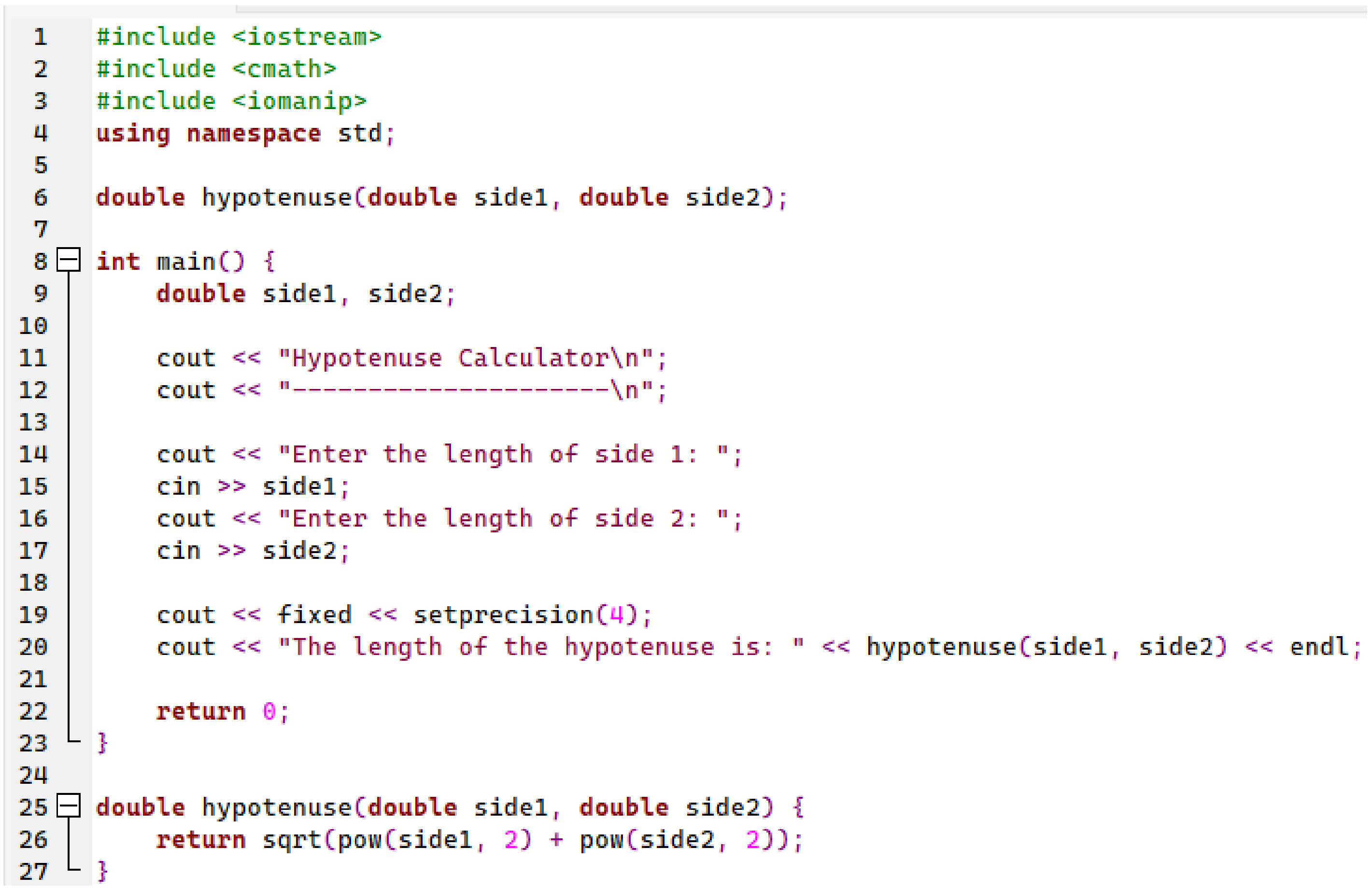
```cpp
#include <iostream>
#include <iomanip>
using namespace std;

float cubeVolume(float side);

int main() {
    float side, result;

    cout << "Cube Volume Calculator\n";
    cout << "------------------------------------------\n";
    cout << "Enter the side length of the cube: ";
    cin >> side;

    result = cubeVolume(side);
    cout << fixed << setprecision(2);
    cout << "The volume of the cube is: " << result << endl;

    return 0;
}

float cubeVolume(float side) {
    float volume;
    volume = side * side * side;
    return volume;
}
```

OUTPUT



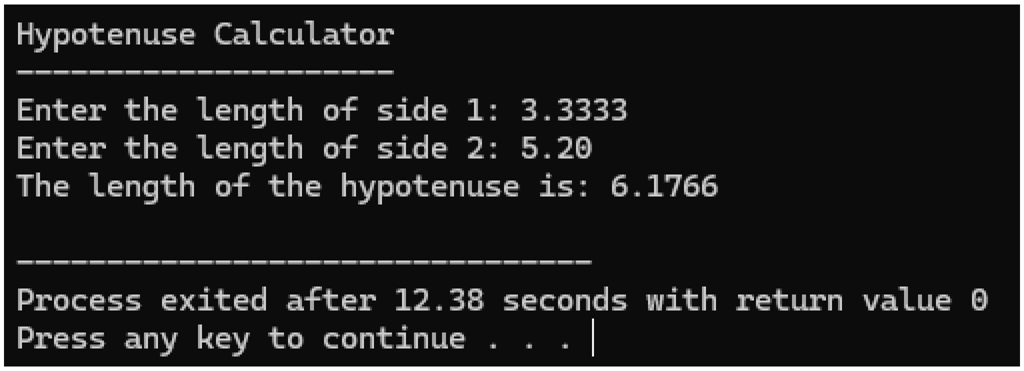
```
Cube Volume Calculator
------------------------------------------
Enter the side length of the cube: 3.475
The volume of the cube is: 41.96

--------------------------------
Process exited after 2.689 seconds with return value 0
Press any key to continue . . .
```

#2
CODE



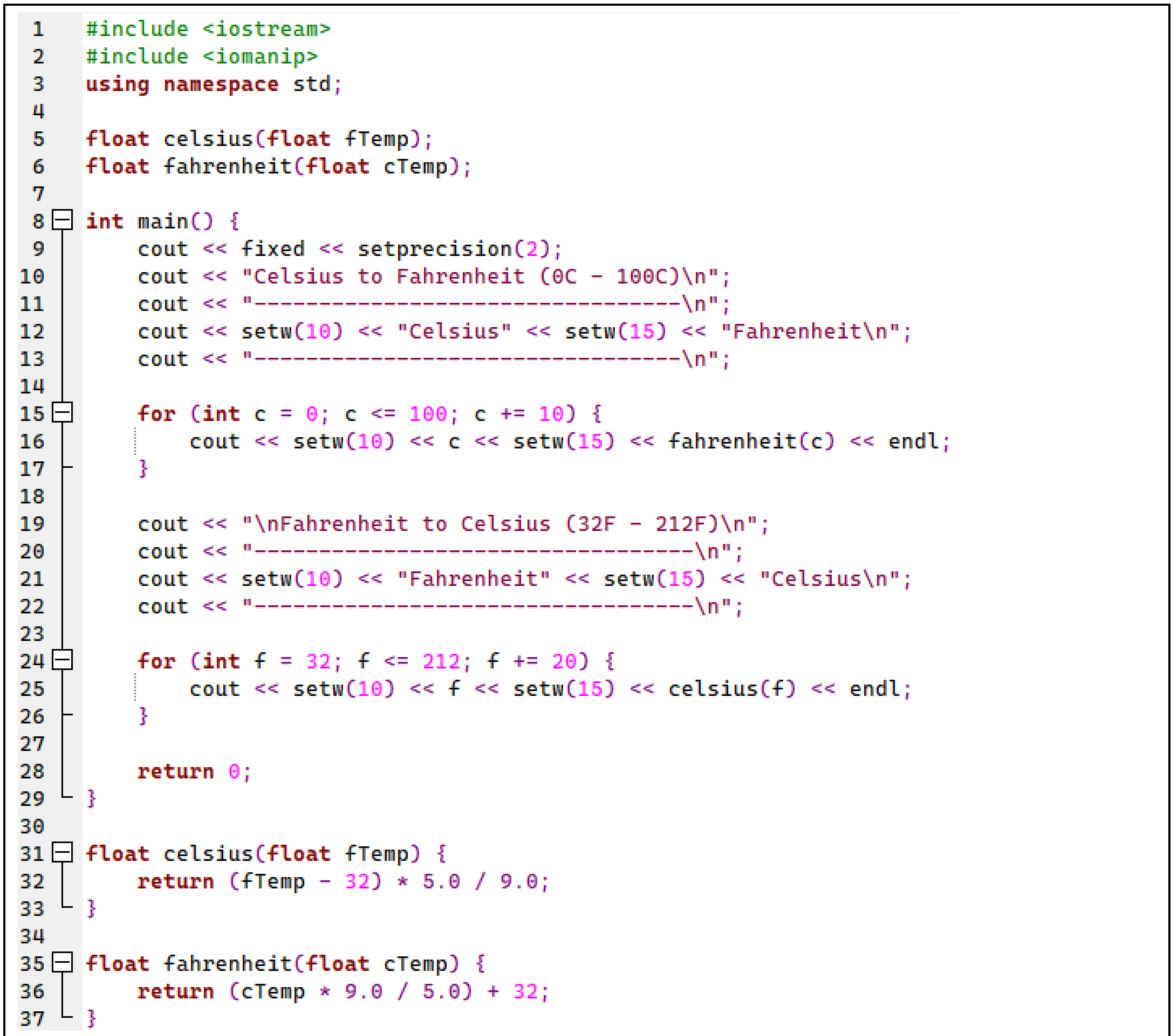
```cpp
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

double hypotenuse(double side1, double side2);

int main() {
    double side1, side2;

    cout << "Hypotenuse Calculator\n";
    cout << "---------------------\n";

    cout << "Enter the length of side 1: ";
    cin >> side1;
    cout << "Enter the length of side 2: ";
    cin >> side2;

    cout << fixed << setprecision(4);
    cout << "The length of the hypotenuse is: " << hypotenuse(side1, side2) << endl;

    return 0;
}

double hypotenuse(double side1, double side2) {
    return sqrt(pow(side1, 2) + pow(side2, 2));
}
```

OUTPUT



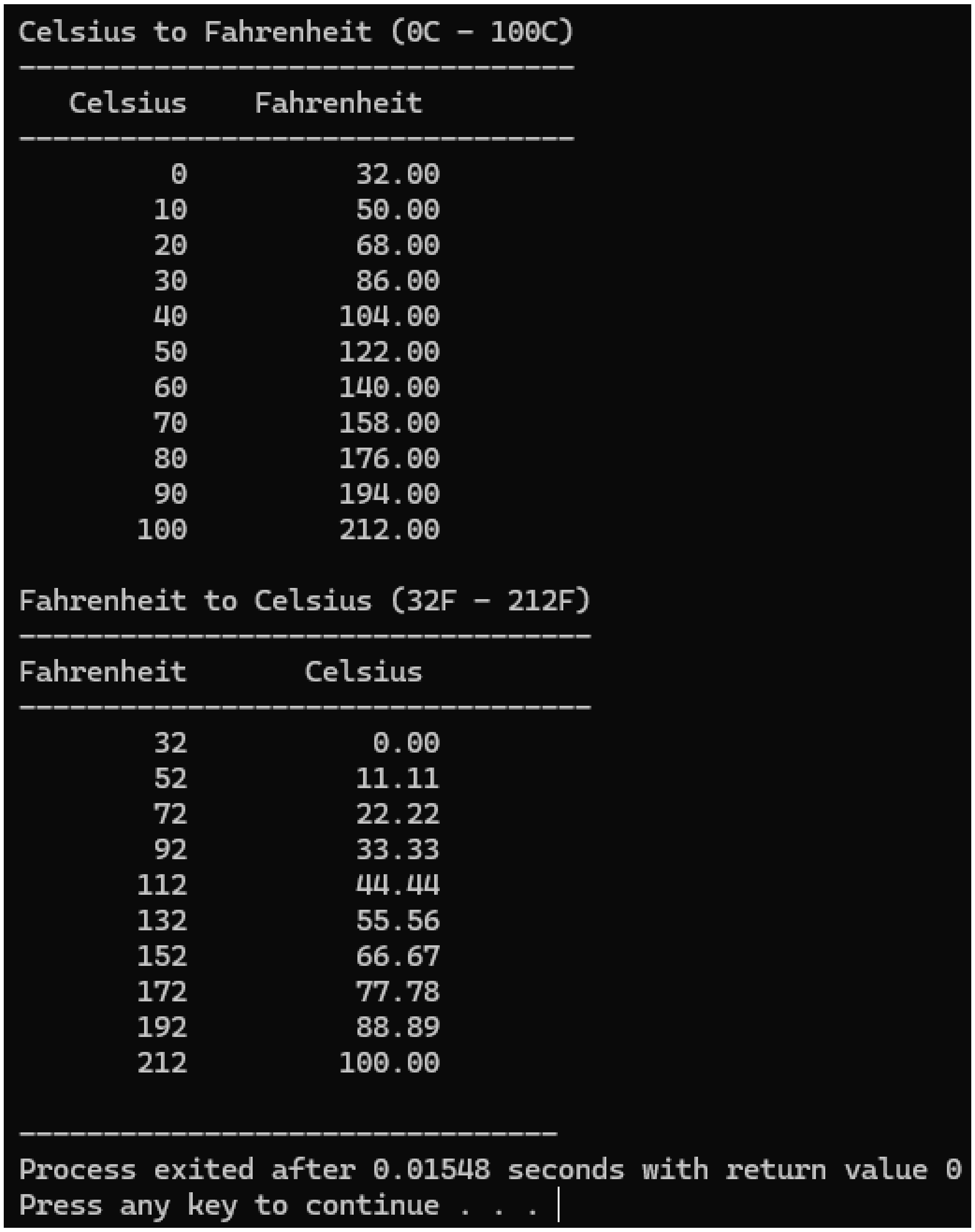
```
Hypotenuse Calculator
---------------------
Enter the length of side 1: 3.3333
Enter the length of side 2: 5.20
The length of the hypotenuse is: 6.1766

--------------------------------
Process exited after 12.38 seconds with return value 0
Press any key to continue . . .
```

#3
CODE



```cpp
#include <iostream>
#include <iomanip>
using namespace std;

float celsius(float fTemp);
float fahrenheit(float cTemp);

int main() {
    cout << fixed << setprecision(2);
    cout << "Celsius to Fahrenheit (0C - 100C)\n";
    cout << "---------------------------------\n";
    cout << setw(10) << "Celsius" << setw(15) << "Fahrenheit\n";
    cout << "---------------------------------\n";

    for (int c = 0; c <= 100; c += 10) {
        cout << setw(10) << c << setw(15) << fahrenheit(c) << endl;
    }

    cout << "\nFahrenheit to Celsius (32F - 212F)\n";
    cout << "----------------------------------\n";
    cout << setw(10) << "Fahrenheit" << setw(15) << "Celsius\n";
    cout << "----------------------------------\n";

    for (int f = 32; f <= 212; f += 20) {
        cout << setw(10) << f << setw(15) << celsius(f) << endl;
    }

    return 0;
}

float celsius(float fTemp) {
    return (fTemp - 32) * 5.0 / 9.0;
}

float fahrenheit(float cTemp) {
    return (cTemp * 9.0 / 5.0) + 32;
}
```

OUTPUT



```
Celsius to Fahrenheit (0C - 100C)
---------------------------------
   Celsius    Fahrenheit
---------------------------------
         0         32.00
        10         50.00
        20         68.00
        30         86.00
        40        104.00
        50        122.00
        60        140.00
        70        158.00
        80        176.00
        90        194.00
       100        212.00

Fahrenheit to Celsius (32F - 212F)
----------------------------------
Fahrenheit       Celsius
----------------------------------
        32          0.00
        52         11.11
        72         22.22
        92         33.33
       112         44.44
       132         55.56
       152         66.67
       172         77.78
       192         88.89
       212        100.00

--------------------------------
Process exited after 0.01548 seconds with return value 0
Press any key to continue . . .
```

7. Supplementary Activity

ANALYSIS

#1

When executing this code, two libraries iostream and iomanip are used. iostream for the input and output, iomanip for fixing the precision point of the floating point values to 4, and using namespace std for cin and cout. Line 5 declares a float function called cubeVolume with the parameters float side. This allows defining the function after main. The main function at line 7 is the starting point of the program where it declares the variables side and result. Line 10 and 11 prints out a header label. Line 12 and 13 prompts and waits for the user input of numbers. Line 15 calls the cubeVolume function and brings the value side of the function, which returns the stored resultant cube volume. Line 16 and 17 outputs the result with the fixed precision. The defined function at line 22 contains the computations. It defines the function with cubeVolume and parameter size based on the user input. It declares its variable volume with float. It then computes volume with side * side * side and returns to pass it back to main.

#2

When executing this code, three libraries iostream, iomanip, and cmath are used for the commands used in the code. A function declaration at line 6 contains the double datatype for hypotenuse function. It contains the parameters double

side1, double side2) for the two inputs later on in this code. The main function starting at line 8 declares two double variables side1 and side2. A header title is printed at line 12 and 13. A user input prompt from line 14 to 17 waits for user input of the length of each side At line 19 to 20, an output of the hypotenuse values calls from the defined function below for the results. The iomanip command fixed << setprecision(4) outputs the value in 4 decimal points. A function definition at line 25 contains the declared function hypotenuse with the parameters double side1, double side2. Using the cmath library, this function computes the value with pow(side1, 2) and pow(side2, 2), which then goes through the sqrt() process to calculate the resultant hypotenuse. This returns the value to function main with hypotenuse(side1, side2).
#3
When executing this code, line 5 and 6 declares the two floating point functions for celsius and fahrenheit with the respective parameters fTemp and cTemp also using the float datatype. This is a declaration for the two functions to be used later in the definitions at the code below the main function. Inside the int main function starting from line 8 to 29, a fixed two decimal precision output using the iomanip library is declared to output the values in a cleaner and more consistent layout. Line 10 prints a title for the table for Celsius to Fahrenheit (0C - 100C), then line 11 prints out a line header for organizing the appearance of the table. Line 12 prints the label for celsius and fahrenheit and separates them using the setw() command from iomanip, and then uses a new line operator to skip over to the next line, which line 13 then prints a header line with the - symbol to organize the table. A for loop at line 15 declares its own loop parameters for celsius with c = 0 for the starting point, c <= 100 for the limit, and c += 10 for 10 step increments. Line 16 inside the loop uses the set() function for organizing the output of c and the fahrenheit(c) function call containing the conversion process below int main. At line 19-22, another table similar to the previous celsius label outputs the title for Fahrenheit to Celsius (32F - 212F), using the same setw() functions and separators with multiple - symbols. Another for loop at line 24 uses its own loop parameters for fahrenheit with f = 32 as the starting point, f <= 212 as the end point, and f += 20 as the increment steps. Line 25 inside the loop outputs the incremented f values and returned values called from celsius(f). Starting from line 31, there are two function definitions celsius and fahrenheit containing the computations and conversions for the temperatures. For celsius, it contains the formula (fTemp - 32) * 5.0 / 9.0 for conversion and returns the value to the function call at main. A similar process at fahrenheit uses this formula (cTemp * 9.0 / 5.0) + 32 and returns it to the function call at main as well.

8. Conclusion

Concluding this activity, I understood that built-in functions are a different type of functions or commands that call for other libraries like iomanip, cmath, and cctype to call for commands and functions like setw(), cin.getline(), and in this activity, hypotenuse(), sqrt(), and pow(). These functions work by calling the libraries for the functions needed to execute the commands properly and using parameters to declare and use variables, and set a value for a command to output spaces like setw(). There are similarities between built-in functions and regular functions in that they call for other sources consisting of blocks of code to execute commands inside their respective names For example, regular functions inside int main call for the outside functions to do certain commands without interfering with the commands in main, while built-in functions call for commands from other libraries mentioned earlier. By using built-in functions, it allows for a more modular and configurable code to do more things, reduce value inaccuracies and errors, and overall, have a cleaner code.