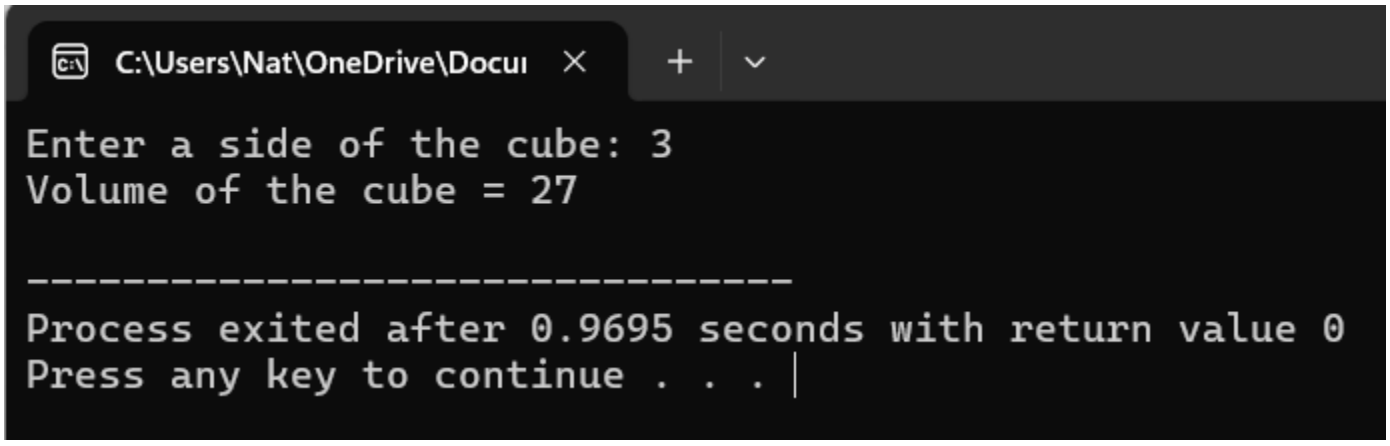
Hands-on Activity 6.2	
Built-in Functions	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed:** October 24, 2025
Section: CPE11S1	**Date Submitted:** October 28, 2025
Name(s): Mendoza, Nathaniel B.	**Instructor: Engr. Jimlord M. Quejado**

6. Output

Code :

```
#include <iostream>
#include <cmath> // for pow()
using namespace std;

int main() {
    float side;
    cout << "Enter the side length of the cube: ";
    cin >> side;

    // Using built-in pow() function: V = s³
    float volume = pow(side, 3);

    cout << "Volume of the cube = " << volume << endl;
    return 0;
}
```

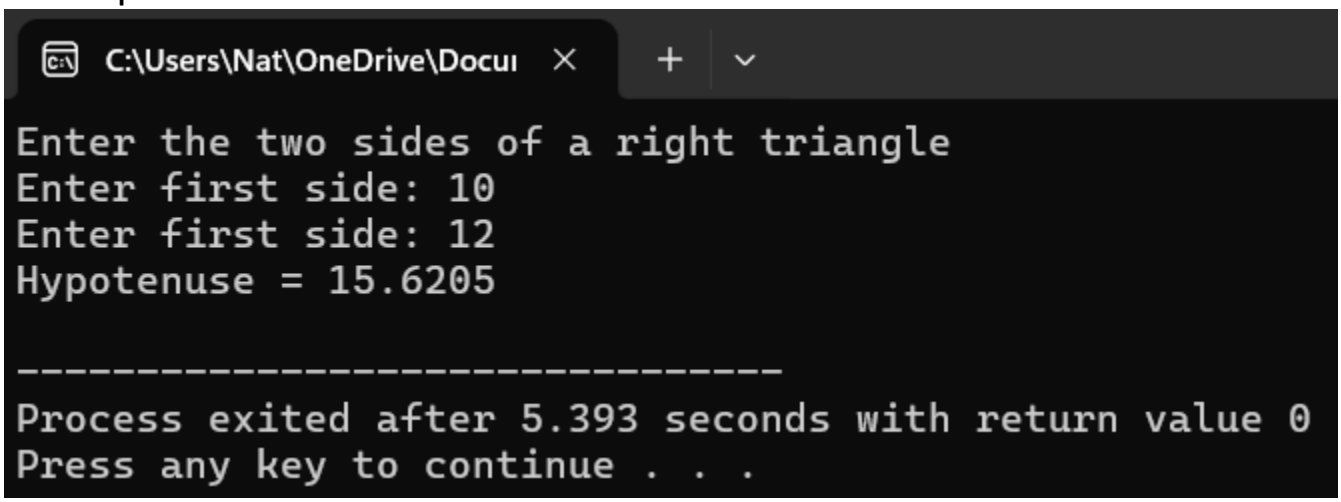
Code Output :



Code :

```
#include <iostream>
#include <cmath> // for sqrt() and pow()
using namespace std;

int main() {
    float side1, side2;
    cout << "Enter the two sides of a right triangle\n";
    cout << "Enter first side: ";
    cin >> side1;
    cout << "Enter first side: ";
            cin >> side2;

    float hypotenuse = sqrt(pow(side1, 2) + pow(side2, 2));

    cout << "Hypotenuse = " << hypotenuse << endl;
    return 0;
}
```

Code Output :

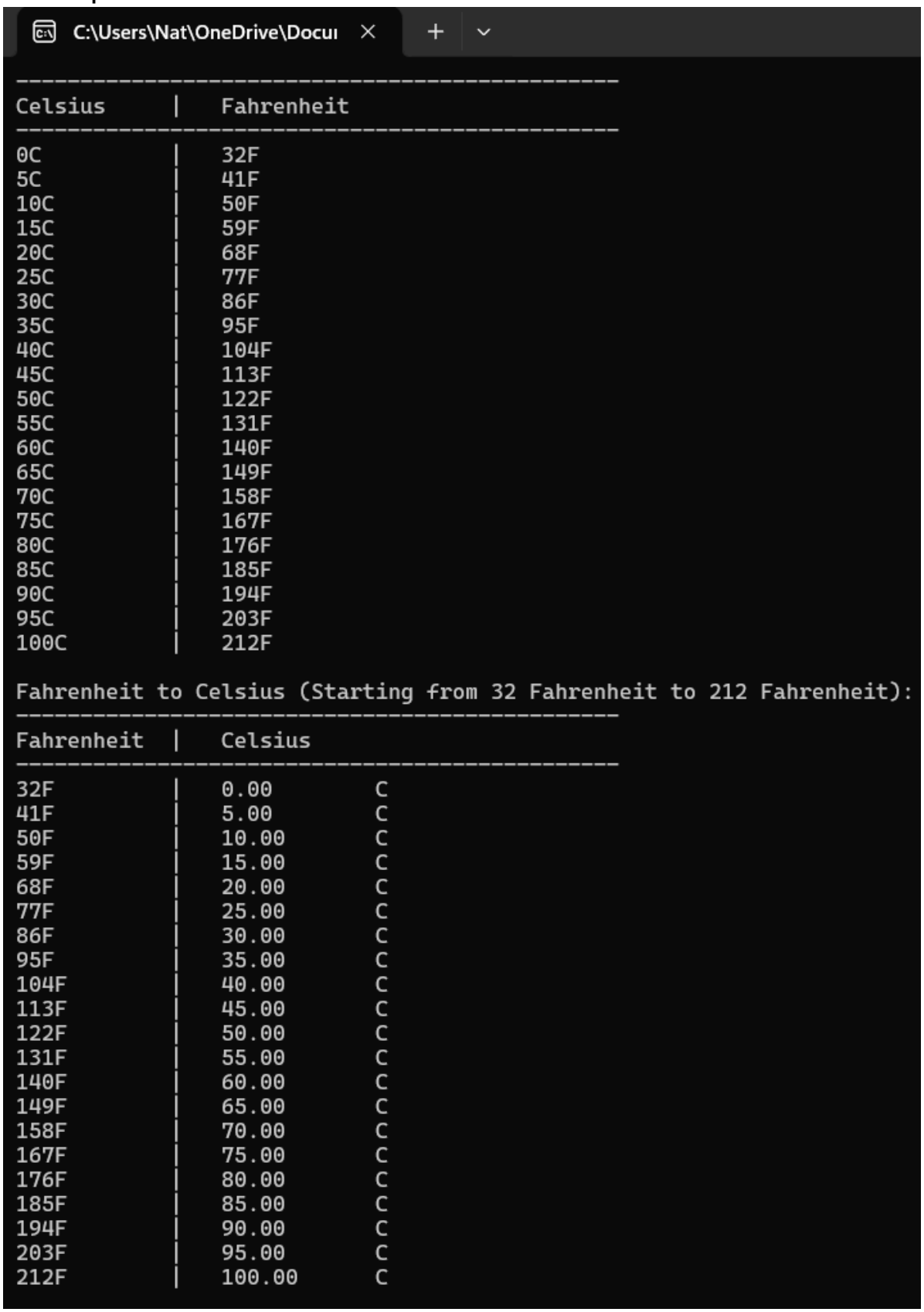


Code :

```
#include <iostream>
#include <iomanip> // for setw and setprecision
using namespace std;

/* Declarations */
void line();
void showCelsiusTable();
void showFahrenheitTable();
void temperatureComp(int operation, float temperature);

int main() {
    cout << "================================================\n";
    cout << "      TEMPERATURE CONVERSION TABLE PROGRAM     \n";
```

```
    cout << "==============================================\n\n";

    // Calling functions
    showCelsiusTable();
    cout << endl;
    showFahrenheitTable();

    return 0;
}

/* Definitions */

void line() {
    cout << "----------------------------------------------\n";
}

void showCelsiusTable() {
    cout << "Celsius to Fahrenheit (Starting from 0 Celsius to 100 Celsius):\n";
    line();
    cout << left << setw(12) << "Celsius" << "|\t" << setw(12) << "Fahrenheit" << endl;
    line();
    temperatureComp(1, 0);
}

void showFahrenheitTable() {
    cout << "Fahrenheit to Celsius (Starting from 32 Fahrenheit to 212 Fahrenheit):\n";
    line();
    cout << left << setw(12) << "Fahrenheit" << "|\t" << setw(12) << "Celsius" << endl;
    line();
    temperatureComp(2, 32);
}

void temperatureComp(int operation, float temperature) {
    if (operation == 1) {
        for (int c = 0; c <= 100; c += 5) {
            float f = (9.0 / 5.0) * c + 32;
            cout << left << setw(12) << (to_string(c) + "C")
                 << "|\t" << setw(12) << (to_string((int)f) + "F") << endl;
        }
    }
    else if (operation == 2) {
        for (int f = 32; f <= 212; f += 9) {
            float c = (5.0 / 9.0) * (f - 32);
            cout << left << setw(12) << (to_string(f) + "F")
                 << "|\t" << setw(12) << fixed << setprecision(2) << c << "C" << endl;
        }
    }
}
```

Code Output :


C:\Users\Nat\OneDrive\Docu
Celsius | Fahrenheit
0C | 32F
5C | 41F
10C | 50F
15C | 59F
20C | 68F
25C | 77F
30C | 86F
35C | 95F
40C | 104F
45C | 113F
50C | 122F
55C | 131F
60C | 140F
65C | 149F
70C | 158F
75C | 167F
80C | 176F
85C | 185F
90C | 194F
95C | 203F
100C | 212F
Fahrenheit to Celsius (Starting from 32 Fahrenheit to 212 Fahrenheit):
Fahrenheit | Celsius
32F | 0.00 C
41F | 5.00 C
50F | 10.00 C
59F | 15.00 C
68F | 20.00 C
77F | 25.00 C
86F | 30.00 C
95F | 35.00 C
104F | 40.00 C
113F | 45.00 C
122F | 50.00 C
131F | 55.00 C
140F | 60.00 C
149F | 65.00 C
158F | 70.00 C
167F | 75.00 C
176F | 80.00 C
185F | 85.00 C
194F | 90.00 C
203F | 95.00 C
212F | 100.00 C

7. Supplementary Activity

Code Analysis #1 :

I started the program by including the <iostream> and <cmath> libraries. First, the <iostream> library allows the program to perform input and output operations such as displaying messages and getting user input, while <cmath> is used for mathematical functions like pow(). The using namespace std; statement is included so that we can use standard C++ functions without the redundancy of std::.

Inside the main() function, a float variable named side is declared to store the length of the cube's side. The program prompts the user to enter the side length of the cube using cout and reads the input with cin. The formula for the volume of a cube, which is $V = s$^3, is then calculated using the pow() function. The result is stored in the variable volume. Finally, the program displays the computed volume of the cube using cout and ends with a return 0; statement to indicate successful execution.

Code Analysis #2 :

The program begins by including <iostream> and <cmath> libraries. The <iostream> library is used for input and output operations, while <cmath> provides mathematical functions such as sqrt() for square root and pow() for exponentiation. The using namespace std; statement simplifies the code by allowing the use of standard C++ commands directly.

In the main() function, two float variables, side1 and side2, are declared to represent the two shorter sides of a right triangle. The program asks the user to input both side lengths using cout and cin. It then calculates the hypotenuse using the Pythagorean theorem c^2= a^2+b^2 and then it will be square rooted by both sides. This is done with the sqrt() and pow() functions from the <cmath> library. The result is stored in the variable hypotenuse and then displayed using cout.

Code Analysis #3 :

This program starts by including the <iostream> and <iomanip> libraries. The <iostream> library handles input and output operations, while <iomanip> is used to format the output neatly using functions such as setw() and setprecision(). The using namespace std; statement is also used for convenience.

The program uses function declarations and definitions to organize the code clearly. There are four declared functions: line(), showCelsiusTable(), showFahrenheitTable(), and temperatureComp().

The main() function serves as the program's entry point. It displays a title header and calls the two functions showCelsiusTable() and showFahrenheitTable() to generate both temperature conversion tables.

The line() function simply prints a line separator (----------) to make the table more readable.

The showCelsiusTable() function prints the Celsius-to-Fahrenheit conversion table. It starts by displaying a header, calls line() for formatting, and then calls temperatureComp(1, 0) to perform the conversion from Celsius to Fahrenheit.

The showFahrenheitTable() function works similarly but converts Fahrenheit to Celsius by calling temperatureComp(2, 32).

The temperatureComp() function contains the actual logic for conversion. If the operation is 1, it runs a for loop from 0°C to 100°C, increasing by 5°C, and converts each value to Fahrenheit using the formula

F=(9/5)∗C+32. If the operation is 2, it loops from 32°F to 212°F, increasing by 9°F, and converts each value to Celsius using the formula
. Each result is then printed neatly in aligned columns using setw() and left formatting.

8. Conclusion

After performing this activity I enhanced and gained more knowledge throughout theI learning. The use of functions in a program makes it more organized, readable, and easier to manage. Functions help separate each part of the program according to its purpose, which makes finding and fixing errors simpler. Instead of writing the same code again and again, I can just create one function and call it whenever I need it. Next is that the most difficult part of the activity is the printing in the creation of the table, it requires proper lining and setting spaces for better precisions of spaces for a better look for the output. Therefore, after doing number 3, I want to improve my skills throughout making the table because this saves time and makes the program look cleaner. I faced a lot of errors while trying to make it work properly, and I really had to look for examples to guide me in fixing the mistakes. Through this process, I realized that coding is never easy nor smooth and simple, but solving those errors step by step taught me patience and persistence. It also showed me that using references and learning from sources is part of the process of becoming better at programming. Through this activity, I also understood the importance of programming where each function handles a specific task, such as mathematics computation and temperature conversion. It made my program more efficient and flexible. Overall, functions are very useful in programming because they make the code reusable, easier to understand, and more professional. Therefore after performing this activity, I realized how useful functions are in simplifying and organizing my code. Learning how to apply them will help me in future programming activities because I can now create cleaner, reusable, and more efficient programs with less effort. Additionally, I found this activity very beneficial and helpful as my team is going to be working on a project in which we are going to apply CRUD which is to create, read, update, and delete. I can say that with the usage of functions this project is going to be less stressful and difficult. This activity made me appreciate how functions can be applied in both simple computations and more complex, interactive programs. I also learned the importance of breaking down problems into smaller steps, like filling up the multiplication table with loops.