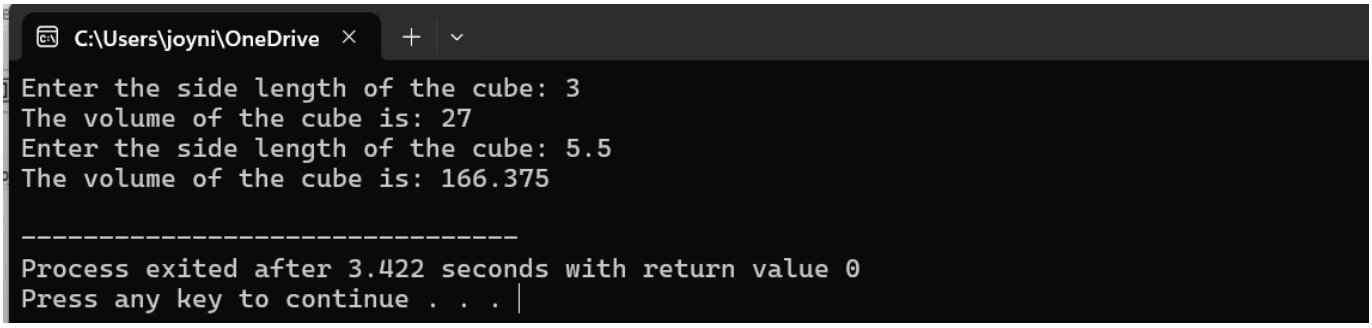
Activity No. 6.2	
Hands-On Activity Built Function	
Course Code: CPE 007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed: 10/23/25**
Section: CPE11S1	**Date Submitted: 10/30/25**
Name(s):Cenndy M. Nieles	**Instructor: Engr. Jimlord M. Quejado**
6. Output	

INPUT:

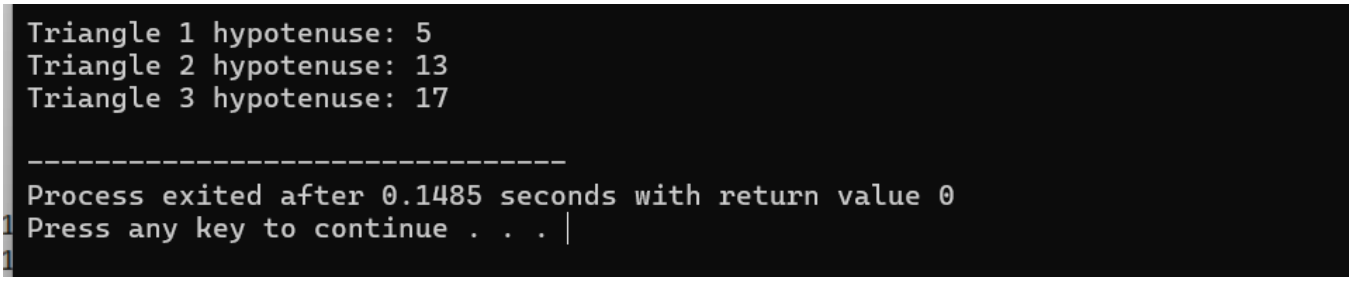
VOLUMECUBE.cpp

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

double  cubeVolume(double s) {
    return s* s* s;
}

    void computeAndDisplayVolume() {
    double side;
    cout << "Enter the side length of the cube: ";
    cin >> side;
    double volume = cubeVolume(side);
    cout << "The volume of the cube is: "<< volume << endl;

}

   int main(){
    computeAndDisplayVolume();
    computeAndDisplayVolume();

    return 0;
   }
```



In conclusion, this program uses a simple built-in arithmetic operation in C++ to calculate the volume of a cube. The function volume takes the side length as input and computes the cube's volume using the formula (V = s * s * s). The program demonstrates how built-in mathematical operations can be used inside a user-defined function to perform accurate calculations. By entering the cube's side length, the user can quickly obtain the volume without complex code. This shows how C++ built-in functionality allows for clear, efficient, and practical problem-solving.

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

double hypotenuse(double side1, double side2) {
    return sqrt(side1 * side1 + side2 * side2);
}

int main() {
    double a1 = 3.0, b1 = 4.0;
    double a2 = 5.0, b2 = 12.0;
    double a3 = 8.0, b3 = 15.0;

    cout << "Triangle 1 hypotenuse: " << hypotenuse(a1, b1) << endl;
    cout << "Triangle 2 hypotenuse: " << hypotenuse(a2, b2) << endl;
    cout << "Triangle 3 hypotenuse: " << hypotenuse(a3, b3) << endl;

    return 0;
}
```



This C++ program calculates the hypotenuse of a right triangle using a built-in math function. It includes the <iostream> and <cmath> headers. The <iostream> header allows the program to perform input and output using cout, while <cmath> provides access to built-in mathematical functions such as sqrt() and pow(). The statement using namespace std; makes it possible to use these features without prefixing them.The function hypotenuse is a user-defined function that calls the C++ built-in function sqrt(). This built-in function computes the square root of a number. Inside hypotenuse, the expression sqrt(side1 * side1 + side2 * side2) applies the Pythagorean theorem to find the hypotenuse. Each side is squared, the results are added, and the square root of the sum is taken using sqrt(). The function then returns the result as a double.

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

// Function to convert Celsius to Fahrenheit
double celsiusToFahrenheit(double celsius) {
    return (9.0 / 5.0) * celsius + 32;
}

// Function to convert Fahrenheit to Celsius
double fahrenheitToCelsius(double fahrenheit) {
    return (5.0 / 9.0) * (fahrenheit - 32);
}

int main() {
    cout << fixed << setprecision(1);
    cout << "Celsius   Fahrenheit | Fahrenheit   Celsius" << endl;
    cout << "-----------------------------------------" << endl;

    int c = 0;
    int f = 32;

    // Print both tables side by side
    while (c <= 100 && f <= 212) {
        cout << setw(6) << c
             << setw(12) << celsiusToFahrenheit(c)
             << " | "
             << setw(8) << f
             << setw(12) << fahrenheitToCelsius(f)
             << endl;

        c += 10;
        f += 18;
    }
```

```
Celsius   Fahrenheit | Fahrenheit   Celsius
-----------------------------------------
     0         32.0 |        32         0.0
    10         50.0 |        50        10.0
    20         68.0 |        68        20.0
    30         86.0 |        86        30.0
    40        104.0 |       104        40.0
    50        122.0 |       122        50.0
    60        140.0 |       140        60.0
    70        158.0 |       158        70.0
    80        176.0 |       176        80.0
    90        194.0 |       194        90.0
   100        212.0 |       212       100.0

--------------------------------
Process exited after 0.1527 seconds with return value 0
Press any key to continue . . .
```

The first function, celsius, converts a Fahrenheit temperature to its Celsius equivalent. It takes an integer argument, subtracts 32 from it, multiplies the result by 5, divides by 9, and adds 0.5 to round the result before returning it as an integer. The second function, fahrenheit, converts a Celsius temperature to its Fahrenheit equivalent. It takes an integer argument, multiplies it by 9, and adds 0.5 to round the result before returning it as an integer. The second function, fahrenheit, converts a Celsius temperature to its Fahrenheit equivalent. It takes an integer argument, multiplies it by 9, divides by 5, adds 32, and adds 0.5 to round the result before returning it as an integer.In the main function, the program prints table headers for both Celsius-to-Fahrenheit and Fahrenheit-to-Celsius conversions. It uses setw() from the <iomanip> library to align the output in columns, making the chart readable. A line of dashes separates the headers from the data. The for loop runs two variables, c and f, where c starts at 0 (representing Celsius) and f starts at 32

(representing Fahrenheit). The loop continues as long as either c is less than or equal to 100 or f is less than or equal to 212.

7. Supplementary Activity

8. Conclusion

This program determines the length of a right triangle's hypotenuse, and uses the hypotenuse() function. The lengths of the other two triangle sides are represented by the two arguments side1 and side2, which are passed to the function.These functions allow the programmer to perform certain common mathematical calculations involving predefined functions such as sqrt(), pow(), sin(), cos(), and log() found in the header. Function arguments may be constants, variables, or expressions called hypotenuse() to determine and show a right triangle's hypotenuse. The function does a task but does not return a value because it has a void return type It requires two parameters, side1 and side2, which stand for the triangle's two perpendicular sides' lengths. Using the function calculates the square root of the sum of the squares of the two sides using the built-in sqrt() function from the library.CelsiusToFahrenheit() and fahrenheitToCelsius(), to convert temperatures between Celsius and Fahrenheit. Each function performs one conversion using standard temperature conversion formulas; the celsiusToFahrenheit() function multiplies the Celsius value by 9.0/5.0 and adds 32 to get the Fahrenheit equivalent. The fahrenheitToCelsius() function subtracts 32 from the Fahrenheit value and multiplies the result by 5.0/9.0 to get the Celsius equivalent. For clarity, it prints a dividing line and a table header a while loop is then used to print the two conversions side by side after initializing two variables, c for Celsius and f for Fahrenheit row displaying a Celsius value along with its Fahrenheit equivalent and a Fahrenheit value along with its Celsius equivalent is printed in each iteration. The setw() function aligns the output neatly into columns. To progress through the temperature ranges, c and f increase by 10 and 18 respectively after each row. After displaying every conversion, the program terminates, demonstrating how built functions make the code accurate, modular, and maintainable.