Hands-on Activity 6.2	
Built-in Functions	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed: 30/10/2025**
Section: CPE11S1	**Date Submitted: 30/10/2025**
Name(s): Ralph Angelov F. Braganza	**Instructor: Engr. Jimlord M. Quejado**

Output

1. Create a program that defines a function to compute for the volume of a cube. The formula of the volume of the cube is given as V = s * s * s.

CODE:

```
#include <iostream>
#include <cmath>
#include <cctype>

float cubeVolume(float side) {
    return std::pow(side, 3);
}

int main() {
    float side;
    float volume;
    char choice;
    bool continueLoop = true;

    while (continueLoop) {
        while (true) {
            std::cout << "\nEnter the side length of the cube: ";

            if (std::cin >> side) {
                if (side > 0) {
                    break;
                } else {
                    std::cout << "Side length must be a positive number.\n" << std::endl;
                }
            } else {
                std::cout << "Invalid input. Please enter a valid number.\n" << std::endl;
                std::cin.clear();
                std::cin.ignore(10000, '\n');
            }
        }

        volume = cubeVolume(side);

        std::cout << "Volume of the cube = " << volume << std::endl;

        while (true) {
            std::cout << "\nDo you want to continue (Y/N)?: ";
```

```
        std::cin >> choice;
        std::cin.ignore(10000, '\n');

        choice = std::toupper(choice);

        if (choice == 'N') {
            continueLoop = false;
            std::cout << "Program exited.";
            break;
        } else if (choice == 'Y') {
            break;
        } else {
            std::cout << "Invalid choice. Please enter Y or N." << std::endl;
        }
    }
  }

  return 0;
}
```

RESULTS (ALL):

```
Enter the side length of the cube: 6
Volume of the cube = 216

Do you want to continue (Y/N)?: y

Enter the side length of the cube: no
Invalid input. Please enter a valid number.


Enter the side length of the cube: -7
Side length must be a positive number.


Enter the side length of the cube: 7
Volume of the cube = 343

Do you want to continue (Y/N)?: maybe
Invalid choice. Please enter Y or N.

Do you want to continue (Y/N)?: n
Program exited.
--------------------------------
Process exited after 36.95 seconds with return value 0
Press any key to continue . . .
```

2. Define a function hypotenuse that calculates the length of the hypotenuse of a right triangle when the other two sides are given. Use this function in a program to determine the length of the hypotenuse for each of the following triangles. The function takes two arguments of type double and return the hypotenuse as a double.

CODE:

```
#include <iostream>
#include <cmath>
#include <cctype>

float calculateHypotenuse(float side1, float side2) {
  return std::sqrt(std::pow(side1, 2) + std::pow(side2, 2));
}
```

```
int main() {
    float side1;
    float side2;
    char choice;
    bool continueLoop = true;

    while (continueLoop) {
        std::cout << "\n----- Right Triangle Calculator -----\n";

        while (true) {
            std::cout << "\nEnter the length of the first side: ";
            if (std::cin >> side1) {
                if (side1 > 0) {
                    break;
                } else {
                    std::cout << "Side length must be a positive number." << std::endl;
                }
            } else {
                std::cout << "Invalid input. Please enter a valid number." << std::endl;
                std::cin.clear();
                std::cin.ignore(10000, '\n');
            }
        }

        while (true) {
            std::cout << "\nEnter the length of the second side: ";
            if (std::cin >> side2) {
                if (side2 > 0) {
                    break;
                } else {
                    std::cout << "Side length must be a positive number." << std::endl;
                }
            } else {
                std::cout << "Invalid input. Please enter a valid number." << std::endl;
                std::cin.clear();
                std::cin.ignore(10000, '\n');
            }
        }

        float hypotenuse = calculateHypotenuse(side1, side2);
        std::cout << "Hypotenuse = " << hypotenuse << std::endl;

        while (true) {
            std::cout << "\nDo you want to calculate another hypotenuse (Y/N)?: ";
            std::cin >> choice;
            std::cin.ignore(10000, '\n');

            choice = std::toupper(choice);

            if (choice == 'N') {
                continueLoop = false;
```

```
            break;
        } else if (choice == 'Y') {
            break;
        } else {
            std::cout << "Invalid choice. Please enter Y or N." << std::endl;
        }
    }
  }

  return 0;
}
```

```
----- Right Triangle Calculator -----

Enter the length of the first side: 6

Enter the length of the second side: 7
Hypotenuse = 9.21954

Do you want to calculate another hypotenuse (Y/N)?: y

----- Right Triangle Calculator -----

Enter the length of the first side: three
Invalid input. Please enter a valid number.

Enter the length of the first side: -3
Side length must be a positive number.

Enter the length of the first side: 3

Enter the length of the second side: for
Invalid input. Please enter a valid number.

Enter the length of the second side: -4
Side length must be a positive number.

Enter the length of the second side: 4
Hypotenuse = 5

Do you want to calculate another hypotenuse (Y/N)?: I think
Invalid choice. Please enter Y or N.

Do you want to calculate another hypotenuse (Y/N)?: n
```

3. Implement the following integer functions:

1. ***Function celsius returns the Celsius equivalent of a Fahrenheit temperature.***
2. ***Function fahrenheit returns the Fahrenheit equivalent of a Celsius temperature.***
3. ***Use these functions to write a program that prints charts showing the Fahrenheit equivalents of a Celsius temperatures from 0 to 100 degrees, and the Celsius equivalents of all Fahrenheit temperatures from 32 to 212 degrees. Print the outputs in a neat tabular format that minimizes the number of lines of output while remaining readable.***

CODE:

```
#include <iostream>
#include <iomanip>

double celsiusToFahrenheit(int celsius)
{
  return (static_cast<double>(celsius) * 9.0 / 5.0) + 32.0;
}

double fahrenheitToCelsius(int fahrenheit)
{
  return (static_cast<double>(fahrenheit) - 32.0) * 5.0 / 9.0;
}
```

```
void printCtoFChart()
{
   std::cout << "---------------------------------" << std::endl;
   std::cout << "|    CELSIUS    |   FAHRENHEIT   |" << std::endl;
   std::cout << "---------------------------------" << std::endl;

   for (int celsius = 0; celsius <= 100; celsius += 5)
   {
      double fahrenheit = celsiusToFahrenheit(celsius);

      std::cout << "| " << std::setw(9) << std::right << celsius << " C"
            << "   | " << std::setw(10) << std::right << std::fixed << std::setprecision(1) << fahrenheit << "F"
            << "    |" << std::endl;
   }
   std::cout << "---------------------------------" << std::endl;
   std::cout << std::endl;
}

void printFtoCChart()
{
   std::cout << "---------------------------------" << std::endl;
   std::cout << "|   FAHRENHEIT  |     CELSIUS    |" << std::endl;
   std::cout << "---------------------------------" << std::endl;

   for (int fahrenheit = 32; fahrenheit <= 212; fahrenheit += 9)
   {
      double celsius = fahrenheitToCelsius(fahrenheit);

      std::cout << "| " << std::setw(9) << std::right << fahrenheit << " F"
            << "   | " << std::setw(9) << std::right << std::fixed << std::setprecision(1) << celsius << " C"
            << "    |" << std::endl;
   }
   std::cout << "---------------------------------" << std::endl;
}

int main()
{
   printCtoFChart();
   printFtoCChart();
   return 0;
}
```

RESULT:

CELSIUS TO FAHRENHEIT → **FAHRENHEIT TO CELSIUS**

CELSIUS	FAHRENHEIT
0 C	32.0F
5 C	41.0F
10 C	50.0F
15 C	59.0F
20 C	68.0F
25 C	77.0F
30 C	86.0F
35 C	95.0F
40 C	104.0F
45 C	113.0F
50 C	122.0F
55 C	131.0F
60 C	140.0F
65 C	149.0F
70 C	158.0F
75 C	167.0F
80 C	176.0F
85 C	185.0F
90 C	194.0F
95 C	203.0F
100 C	212.0F

FAHRENHEIT	CELSIUS
32 F	0.0 C
41 F	5.0 C
50 F	10.0 C
59 F	15.0 C
68 F	20.0 C
77 F	25.0 C
86 F	30.0 C
95 F	35.0 C
104 F	40.0 C
113 F	45.0 C
122 F	50.0 C
131 F	55.0 C
140 F	60.0 C
149 F	65.0 C
158 F	70.0 C
167 F	75.0 C
176 F	80.0 C
185 F	85.0 C
194 F	90.0 C
203 F	95.0 C
212 F	100.0 C

Supplementary Activity

CODE 1 ANALYSIS:

Here I've created a C++ code where it calculates the volume of a cube. First at the very top, we have three libraries. #include <iostream> allows the program to use input (std::cin) and output (std::cout) for user interaction, #include <cmath> provides access to mathematical functions like std::pow() the other functions that were mentioned, which is used to raise a number to a power and the other functions that were mentioned from the activity and #include <cctype> includes character-related functions such as std::toupper(), which converts letters to uppercase which is useful for handling user input like 'y' or 'n' if we want to continue using the program or not.

Now for the function definition, cubeVolume(float side) the function's job is to calculate the volume of a cube.
A cube is a three-dimensional shape with equal sides, so the formula for its volume is:

Volume = s*s*s

So, by implementing std::pow(side, 3), which raises the variable side to the power of 3 this will make things easier instead of manually inputting that number 3 times. It returns the computed volume as a float value to the main function.
The main() function is where the program actually runs and where the variables are declared in the code, volume; will store the computed cube volume, float volume; stores the computed cube volume. char choice; stores the user's decision to continue or stop ('Y' or 'N') and bool continueLoop = true; controls the main loop so the program keeps running until the user decides to quit. The outer while loop (while (continueLoop)) ensures that the entire process will keep asking for an input, computing the volume, and displaying it and it repeats as long as the user wants to continue.
It runs continuously until continueLoop is set to false (when the user enters 'N'). Inside the main loop, there's another while (true) loop for the user's input (meaning if the user actually put a valid number). This then attempts to read the input into the variable side using std::cin >> side;. If the input is valid and positive (side > 0), the loop breaks, allowing the program to proceed. If the user doesn't enter a number value (like a letter), the program detects the failure using the if (!(std::cin >> side)), this will proceed to print an error message "Invalid input. Please enter a valid number." then clears the input error using std::cin.clear() and ignores the remaining invalid characters with std::cin.ignore(10000, '\n') (in other words the repeated spam). If the user enters a negative or zero value, it prints "Side length must be a positive number" and repeats the prompt. Once the user provides a valid number, the program calls calculateCubeVolume.

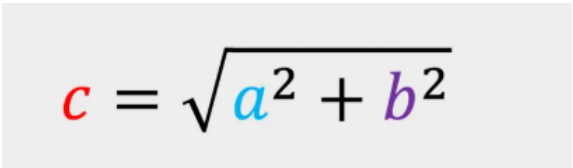
volume = calculateCubeVolume(side);

This passes the user's side length to the earlier-defined function, which computes and returns the cube's volume. The result is then displayed

std::cout << "Volume of the cube = " << volume << std::endl;

After showing the volume, the program asks the user if they want to continue or not (std::cout << "\nDo you want to continue (Y/N)?: ";). The input is stored in choice, and std::toupper(choice) ensures that both lowercase ('y', 'n') and uppercase ('Y', 'N') are treated the same. So if the user enters 'Y', the loop simply restarts, allowing another calculation, if the user enters 'N', continueLoop is set to false, which stops the main loop and ends the program and finally if the user types anything else, an error message appears "Invalid choice. Please enter Y or N." and it will ask again.

CODE 2 ANALYSIS :
In this C++ program I've created a right triangle calculator, it will ask the user to enter the lengths of two sides of a right triangle and then compute the hypotenuse using the Pythagorean theorem:



(credits to chilimath.com for the image)

So first we have our libraries that I've used to make this code functional <iostream> that enables input (std::cin) and output (std::cout), <cmath> to provide mathematical functions such as std::pow() (for squaring) and std::sqrt() (for square roots) and <cctype> for character functions like std::toupper(), which converts letters to uppercase for consistent input handling ('Y', 'N').

Here is our function:

```
float calculateHypotenuse(float side1, float side2) {
    return std::sqrt(std::pow(side1, 2) + std::pow(side2, 2));
}
```

This function computes the hypotenuse given two side lengths, it takes two floating-point numbers (side1 and side2) as inputs then uses the Pythagorean theorem formula and this is possible by using the built-in functions from C++ (std::pow() (for squaring) and std::sqrt()). Now for the int main function where the program starts executing, here we have several variables that are declared. float side1, side2; stores the user's input for the two sides, char choice; stores the user's "Y" or "N" choice and bool continueLoop = true; controls the main loop; when false, the program stops. So the main loop (while (continueLoop)) allows the program to run multiple times, asking for new inputs until the user says they want to stop, it will first ask the user for the first side then goes through the error checking process if the input is either a negative, zero or it isn't a number at all (this ensures the user can't crash the program or enter nonsense data). The same logic is applied for side2.

```
float hypotenuse = calculateHypotenuse(side1, side2);
std::cout << "Hypotenuse = " << hypotenuse << std::endl;
```

Once both sides are valid the function calculateHypotenuse() is called then the returned result (a float) is stored in the variable hypotenuse and the program will print the result to the screen. After showing the result, the program asks if the user wants to perform another calculation. The input (choice) is read and converted to uppercase (so both "y" and "Y" work the same), so if the user enters 'Y' the main loop repeats, allowing another calculation but if 'N' was inputted continueLoop becomes false, breaking the main loop and ending the program and anything other than that prints an error and asks again.

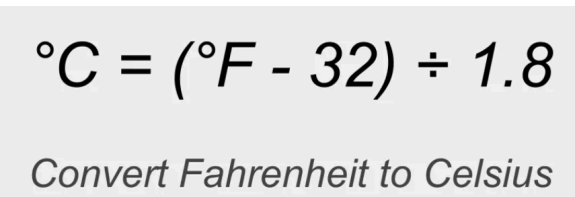
CODE 3 ANALYSIS:

This program will create a Celsius-to-Fahrenheit chart from 0°C to 100°C and a Fahrenheit-to-Celsius chart from 32°F to 212°F. For the header files we have <iostream> which enables input and output (like std::cout for printing) and <iomanip> allows us to control spacing and decimal places using functions like std::setw() and std::setprecision().

Now onto our conversion functions

```
double celsiusToFahrenheit(int celsius)
{
    return (static_cast<double>(celsius) * 9.0 / 5.0) + 32.0;
}
```

This function converts a temperature from Celsius to Fahrenheit using the formula:



(credits to calculatorsoup.com for the image)

The function accepts an integer (celsius) as input, static_cast<double>(celsius) ensures the calculation is done in decimal (floating-point) form, preventing rounding errors and it returns the computed Fahrenheit value as a double.

```
double fahrenheitToCelsius(int fahrenheit)
{
   return (static_cast<double>(fahrenheit) - 32.0) * 5.0 / 9.0;
}
```

This is the section function of code where it converts a temperature from Fahrenheit to Celsius using the formula:

(credits to calculatorsoup.com for the image)

Like before, static_cast<double>(fahrenheit) ensures precise floating-point math then it returns the result as a double. void function printCtoFChart() will print a formatted Celsius-to-Fahrenheit conversion table. So it begins by printing the header then, a for loop generates rows of data.

```
for (int celsius = 0; celsius <= 100; celsius += 5)
```

Here it starts at 0°C and increases by 5°C each time and it will continue to do that until it reaches 100°C. For each value of celsius, it calls celsiusToFahrenheit(celsius) to calculate the Fahrenheit equivalent. After all the results have been calculated the results are printed in a formatted table row with this line of code.

```
std::cout << "| " << std::setw(9) << std::right << celsius << " C"
          << "   | " << std::setw(10) << std::right << std::fixed << std::setprecision(1) << fahrenheit << "F"
          << "    |" << std::endl;
```

std::setw() controls the width of each column for proper alignment then std::setprecision(1) shows one decimal place (for example 97.6°F) and std::fixed ensures the number is printed in standard decimal form (not scientific notation). The output is neatly formatted in columns with the table ending with a closing line (which is just a bunch of negative signs).

Next is of course the function printFtoCChart() which works the same way but in reverse but it prints Fahrenheit-to-Celsius conversions instead. Again, it starts with the table header then the for loop

```
for (int fahrenheit = 32; fahrenheit <= 212; fahrenheit += 9)
```

Starts at 32°F (because that is what the instructions wanted to start with) and increases by 9°F each step until 212°F then it calls fahrenheitToCelsius(fahrenheit) to compute each Celsius value. It then proceeds to print each row with the neat formatting using std::setw() to keep the columns aligned and one decimal place for setprecision and it ends with another line of negative signs (which is for the design of the tabular format). Finally we have our main() function, it first prints the Celsius-to-Fahrenheit chart by calling printCtoFChart() then it prints the Fahrenheit-to-Celsius chart by calling printFtoCChart() and return 0; which is the end of the program.

Conclusion

This activity was quite easy to start with because I already had a foundation of how functions worked and how I could call them back in specific lines of code if they were needed for computation or needed for some parameters. I was able to successfully apply std::pow() for the first program that needed to be coded and was able to utilize it for what it was used for. I do wish, however, I was able to go more in-depth with the other built-in functions, but I guess that will be for another time in the future. What took me the longest was figuring out the spacing because for a couple of minutes I've been just

using the manual space to create the spacing for the tabular formats, and then I remembered the setw() function exists in the <iomanip> library. It took me quite some time to get the spacing right, but I managed to create it with some lazy spacing done for the borders of the Fahrenheit to Celsius and vice versa (it works, but if I had more brain capacity, I would've fixed it a little bit better than that). I'd say I did a pretty good job, and honestly what took me the longest was trying to explain the code as neatly as possible because I use these hands-on activities as reviews, so when I look back to these activities, I can recall what I did and how I was able to come up with my execution for the code. Again, there are still a lot of things I need to improve on to be able to go on to the next level of coding in C++ or any coding language, rather. I hope and pray that in the future I can really just dedicate my time to understanding/mastering coding at a more in-depth level and explore more libraries and built-in functions for me to be able to make my code run more efficiently and look neater.