

Tutorial 6 – Functions (2)

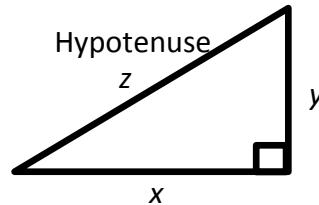
- Q1. Write a function **hypoten()** that computes the length of the hypotenuse of a right-angled triangle after it accepts the lengths of the two opposing sides as function arguments. The function prototype of **hypoten()** is as follow:

```
double hypoten(double, double);
```

Write a program to demonstrate the use of function **hypoten()**. The program should get user's input of the opposing sides, and show the result.

Sample result:

```
Please enter the length of first side: 3
Please enter the length of second side: 4
Hypotenuse of a 3 by 4 right triangle is 5
```



Hints:

- (i) Use a Math library function (in <cmath>) **sqrt()** to calculate a square root value. Its prototype is:
double sqrt(double x);
- (ii) Use Pythagorean Theorem in your calculation: for a right-angled triangle above, $x^2 + y^2 = z^2$, where x and y are the opposing sides, and z is the hypotenuse.

- Q2. Write a function **sumAvg()** that calculates the sum and average of all integers within two input numbers, inclusively. The function takes two integer arguments as the *lower bound* and *upper bound* of the range; a reference to integer argument for storing the *sum* result; and a reference to floating point argument for storing the *average* result. What is the function prototype of **sumAvg()**?

You may assume that the value of lower bound is always smaller than the value of upper bound. The main function below demonstrates the use of the *sumAvg()* function (downloadable from Moodle):

```
int main() {
    int lower, upper, sum;
    float average;

    cout << "Enter the lower bound: ";
    cin >> lower;
    cout << "Enter the upper bound: ";
    cin >> upper;

    sumAvg(lower, upper, sum, average);

    cout << "From " << lower << " to " << upper << ":\n";
    cout << "Sum      = " << sum << endl;
    cout << "Average = " << average << endl;

    return 0;
}
```

Sample result:

```
Enter the lower bound: 5
Enter the upper bound: 10
From 5 to 10:
Sum      = 45
Average = 7.5
```

- Q3. In tutorial 4, you wrote a program that calculates the value of PI using iterative approach (i.e. loop). Now, write a function **calcPI()** that calculates and returns the value of PI **using recursion**. The function takes one integer argument indicating the number of terms used in the calculation. The function prototype of **calcPI()** is as follow:

```
double calcPI(int);
```

Formula of PI: $\text{PI} = 4/1 - 4/3 + 4/5 - 4/7 + 4/9 - 4/11 + 4/13 - \dots$

Examples:

Use 5 terms: $\text{PI} = 4/1 - 4/3 + 4/5 - 4/7 + 4/9$

Use 10 terms: $\text{PI} = 4/1 - 4/3 + 4/5 - 4/7 + 4/9 - 4/11 + 4/13 - 4/15 + 4/17 - 4/19$

Write a program to demonstrate the use of function **calcPI()** and generate the sample result given below. Display the value of PI in 15 decimal places. Note the performance difference between iterative and recursive approaches.

Sample result:

How many terms for PI: **1000**

PI with 1000 terms is 3.140592653839794

- Q4. Write a function **printBinary()** that takes one integer argument **n** and prints the binary version of that number **using recursion**. You may assume that the input argument is always positive. The function prototype of **printBinary()** is as follow:

```
void printBinary(int);
```

Hint: You may consider that each recursive function call only prints the last binary digit of the number.

The main function below demonstrates the use of the **printBinary()** function (downloadable from Moodle):

```
int main() {
    int num;

    cout << "Input a positive decimal integer: ";
    cin >> num;

    cout << "The binary version is ";
    printBinary(num);
    cout << endl;

    return 0;
}
```

Sample result:

Input a positive decimal integer: **123**

The binary version is 1111011