**ICS 103: Computer Programming in C**

**Lab #7: Functions with Output Parameters and InputOutput Parameters**

**Objective:**

* Learn how to write and use functions that return more than one result.

**Functions with output arguments**

In the previous lab, we learned how to write a function that receives arguments (input arguments) and that returns a single result using the return statement. There are many situations where we would like a function to return more than one result. Here are some examples:

* Function to convert time in seconds into hours, minutes and seconds
* Function to find the quotient and remainder of an integer division
* Function to return the maximum, the minimum and the average of a set of values

The return statement cannot be used to return more than one value, i.e. we cannot have the following statement:

return value1, value2;

Also, we cannot have two return statements following each other as:

return value1;

return value2;

The solution in these cases is to make the return type of the function void i.e. the function does not use the return statement to return the results. Instead, the results will be returned through the arguments (output arguments or inputOutput arguments). In contrast to input arguments which are initialized normal variables, constants or expressions, output arguments and inputOutput arguments must be addresses and their corresponding parameters are pointer variables. When pointer variables are used as parameters, they allow the called function to access the variables created in the calling function. Thus, the results can be stored directly in these variables while the execution is in the called function.

Example: Write the analysis, pseudocode algorithm and then translate it into an interactive C program that reads the radius of a circle, it then calls a function that returns the area and the circumference of the circle. The program finally displays the returned values.

**Analysis:**

Input: radius

Input Restriction: radius ≥ 0

Constants: π = 3.14159

Relevant formulas: Area = π \* radius2 , Circumference =2\*π\*radius

Output: Area, Circuference

**Pseudocode:**

Main()

1. Prompt for radius [cm]
2. Input: radius
3. GetAreaAndCircumference(**in:** radius**;** **out**: area, circumference)
4. Output: “Area = ”, **area**, “ sqr cm, Circumference = ”, **circumference**, “ cm”
5. Stop

EndMain

GetAreaAndCircumference(**in:** radius**;** **out:** area, circumference)

1. area = π \* radius2
2. circumference =2\*π\*radius
3. return;

EndGetAreaAndCircumference

The C program is:

**#include <stdio.h>**

**#define PI 3.14159**

**void GetAreaAndCircumference(double radius, double\* area, double\* circum);**

**int main (void)**

**{ double radius, a, c;**

**printf ("Enter the radius of the circle: ");**

**scanf ("%lf", &radius);**

**GetAreaAndCircumference(radius, &a, &c); // function call**

**printf("The area = %f sq cm and circumference is %f cm\n", a, c);**

**return 0;**

**}**

**void GetAreaAndCircumference(double radius, double\* area, double\* circum)**

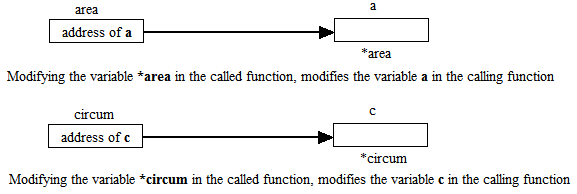
**{ \*area = PI \* radius \* radius;**

**\*circum = 2 \* PI \* radius;**

**return; // optional**

**}**

The function **GetAreaAndCircumference** is to return the area and circumference of a circle (two results) given its radius. Since we want the function to return two results, the return type will be void and we need two output arguments which must be pointer variables (**area** and **circum**). In the calling function (main), we declare two normal variables (**a** for area and **c** for circumference). Notice in the call, the actual arguments are the addresses of the variables **a** and **c**. Once the execution returns to the calling function (main), the values of area and circumference are stored in the variables **a** and **c**.

****

**Input/Output arguments and parameters**

An input/output argument and its corresponding parameter is used to send a value to a called function and then return another value from that function.

**Example:** The pseudocode of a function to swap (or exchange) two values:

**swap(inOut: value1, value2)**

1. **temp = value1**
2. **value1 = value2**
3. **value2 = temp**
4. **return**

**EndSwap**

The C function is:

**void swap(double\* value1, double\* value2){**

**double temp;**

**temp = \*value1;**

**\*value1 = \*value2;**

**\*value2 = temp;**

**return; // optional**

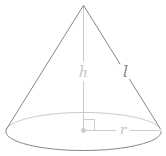
**}**

**Laboratory Tasks:**

**Task1:**

Write a C program that computes the surface area and volume of a cone in square cm and cubic cm respectively. The program has the **main** function and two other functions **getInput** and **computeResults**. The **getInput** function is used to prompt for and read the radius **r** and the length ***l*** of the slant side of the cone both in cm, it then returns these two values.

The **computeResults** function receives the radius **r** and the length ***l*** of the slant side of the cone, it then returns the surface area and the volume of the cone.



**Hint:**

**Note:**

* Your program must define a constant π with a value of 3.141592
* The **computeResults** function must not contain **scanf** or **printf** statements; the printing of the results must be in the **main** function.
* The **main** function must display an appropriate error message and terminate if the input is invalid.

Sample program runs:

|  |
| --- |
|  |
|  |
|  |

Task2:

Write a function **findNumbers** that receives the product, of type **double**, of two numbers x and y (product = x \* y) and the sum, of type **double**, of those two numbers (sum = x + y), it then finds and returns the two numbers x and y, each of type **double**. Write an interactive main function to test your **findNumbers** function. Your program must display an error message and terminate if the input does not satisfy the condition shown below.

**Note:**

* Substitute y=sum-x in the product eq. to get the quadratic equation to be solved for x and y
* Use valid product, sum pairs for your input [sum2-4*\**product >= 0].

**Hint:** You will be solving a quadratic equation of the form: ax2 + bx + c = 0

Use the larger root for your solution:

Sample program runs:

|  |
| --- |
|  |
|  |
|  |

Task3:

Write a function ***sumProductAndNumDigits*** that receives a positive integer value >= 10, it then returns the sum, the product, and the number of the digits of the number. Write an interactive main function to test your function. Your main function must test for input validity. It must loop and display an error message as long as the input is less than 10; otherwise, if the input is valid, it calls the function ***sumProductAndNumDigits*** and displays the returned values.

Note: Your ***sumProductAndNumDigits*** function must not contain any **scanf** or **printf** statements.

Sample program runs:

|  |
| --- |
|  |
|  |
|  |

Task4:

The longitude of a geographic location can be expressed in one the following two formats:

* Decimal degrees. Example: **50.15395**
* Degrees, minutes, and seconds. Example: **50 9 14.220000**

Where **Degrees** and **minutes** are integers, and **seconds** is a value of type **double**.

Write a **two function** C program, in which the **main** function prompts for and reads Decimal degree input. If the input is invalid your program must display an appropriate error message and terminate; otherwise the **main** function passes the input to a void function **toDegreesMinutesAndSeconds** that returns the equivalent degrees, minutes, and seconds. Finally, the **main** function displays the returned values.

**Note**:

* Your program must be general and it must behave as in the sample program runs below.
* The function **toDegreesMinutesAndSeconds** must not have **scanf** and **printf** calls.

**Hint**:

* A longitude can have values from **0.0** degrees to **180.0** degrees inclusive, i.e., **0 ≤ degrees ≤ 180**
* 1 degree = 60 minutes, 1 minute = 60 seconds.

|  |
| --- |
|  |
|  |
|  |