Programming Fundamentals I

COSC1436—Fall 2015

Lab 6. User-Defined Functions

Due Nov. 4, 2015

Objectives:

1. to learn to define functions.
2. to learn to call functions.
3. to learn to pass parameters by value and by reference.

Make sure to demo your work to your instructor for each task to get credit.

Task 1: Defining functions without parameters

Exercise 1. Create a new C++ project in Visual Studio, add a new C++ file and copy the following code into it:

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

void  display();

int main ()
{
    display();
    return 0;
}

//**************************************

void  display()
{
    /* TO BE FILLED IN */
}
```

Exercise 2. Fill in the code in function display() to display your name and address on the screen.

Exercise 3. Build your project and test it.

Demo program to your instructor.

Task 2: Functions with Value Parameters

The following program (see file *starBox.txt*) is the shell of a program that displays the border of a box of starts of size `n` by `n/2` on the screen monitor, where `n` is a positive integer number. For example, if `n` is 11, the following box is displayed:

```
***********
*         *
*         *
*         *
***********
```

The program prompts the user to enter a positive integer number, reads the number and passes it as a parameter to a function `displayBox()` which displays the box on the screen.

```
// displays the border of a box of size n by n/2 where n is a positive integer
// using the star symbol.
#include <iostream>
using namespace std;

void  displayStarBox(int );

int main ()
{
    int  number;

    cout  << "Enter the number of stars for the top; Enter 0 to quit " << endl;
    cin  >> number;

    while (number > 0 )  // this loops allows user to display more than one box of different sizes
    {
       /* WRITE call to displayStarBox() */

       cout  << endl << "Enter the number of stars for the top; Enter 0 to quit " << endl;

       cin >> number;
    }

    system ("pause");
    return 0;
}
void  displayStarBox(int n)
/* 1. ADD documentation: */
// Task:  ??
// Data in:   ??
// Data out:  none
// Data returned: ??
{
    /* 2. ADD code to display n stars on one line for top border line of box*/

    /* 3. ADD code to display (size / 2)-2 lines with a star lining up under the left-most star
          on the top line and a star lining up under the right-most star   */

    /* 4. ADD code to display size stars on one line for bottom border line of box */

}
```

Exercise 1. Add a function call to `displayStarBox()` in function `main()`. Build and run your program.

Exercise 2. Write a brief documentation for the function `displayStarBox()` that describes what the function does.

Exercise 3. Add the missing code in function `displayStarBox()` to display the box. Build and run your project.

Exercise 4. Modify your program so that the symbol used to display the box is also read from the standard input (keyboard) in function `main()` and passed to function `displayStarBox()` as a parameter. Write the steps required to make the symbol a parameter below before modifying your code.

Exercise 5. Test your program with the following input:

10	*
10	$
13	a

Demo your program to your instructor.

Task 3: Functions with Reference Parameters

The program below (see file *wageCalc.txt*) reads two values from the standard input device (i.e. keyboard) `hoursWorked` and `payRate` and outputs the `wage` on the screen. Function `main()` calls function `getData` which prompts the user for the input data values, reads them, and passes them back to `main()` as reference parameters.

```
// Program reads an employee's hourly pay rate and the number of hours worked and displays the
//wage
#include <iostream>
#include <iomanip>
using namespace std;

/* FILL IN the function prototype for getData */
int main ()
{
    float  hoursWorked=0, payRate=0, wage=0;

    cout  << fixed  << showpoint;

    /* WRITE code to call function getData */

    wage = hoursWorked * payRate  ;
    cout  << setw(10)  << hoursWorked
          << setw(10)  << payRate
          << setw(10)  << wage  << endl;

       system("pause");
    return 0;
}
//*******************************************************
/* WRITE the function heading for getData */
// Task:  // prompts the user for hours worked and pay rate, reads them, and passes them back
//        to main() as reference parameters
// Data in:   ??
// Data out:  ??
// Data returned: ??
{
    /* WRITE Code to prompt for hours worked and pay rate */
    /* WRITE Code to read read hours worked and pay rate */
}
```

Exercise 1. Write the heading for function `getData`. Need to determine the (1) return type, (2) parameters passed to functions and their types, and (3) whether parameter need to be passed by value or by reference.

Exercise 2. Write the code to call function `getData` from `main()`.

Exercise 3. Write the function prototype for `getData.` Compile your program and fix any syntax errors (note: since the body of function `getData` is empty, the program doesn't do anything. So the purpose of this run is just fix syntax errors).

Exercise 4. Write the code for the body of function `getData`. Compile and test your program.

Demo your modified program for credit.

Task 4: Value Returning Functions

Consider the program below (see file *currencyConverter.txt*):

```
// This program will input American money and convert it to foreign currency
// PLACE YOUR NAME HERE

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

/* ADD code for function prototypes */
int main ()
{
       float dollars; float euros;  float yens;
       cout << fixed << showpoint << setprecision(2);
       cout << "Please input the amount of American Dollars you want converted "
       << endl;
       cout << "to euros and yen" << endl;
       cin >> dollars;

       // Fill in the code to call currency conversion functions

       // Write output statement(s) to display values in dollar, euro, and yen on same line

       system("pause");
       return 0;
}

float convertToYen(float dollars)
// Task: This function takes a dollar value and converts it to yens
// Data in: dollars
// Data out:  none
// Data returned: equivalent value in yens
{
       cout << "The function convertToYen was called with " << dollars <<" dollars"
       << endl << endl;

       /* FILL IN code to convert value in parameter dollars from dollar to yen.
          Lookup exchange rate on the Internet */

       /*  Add code to return value */

}

float convertToEuros(float dollars)
// Task: This function takes a dollar value and converts it to euros
// Data in: dollars
// Data out:
// Data returned: equivalent value in euros
{

       cout << "The function convertToEuros was called with " << dollars
       << " dollars" << endl << endl;
       /* FILL IN code to convert value in parameter dollars from dollar to yen.
          Lookup exchange rate on the Internet */
       /*  Add code to return value */
}
```

Exercise 1. Complete the code for the above program. Demo it with the following input:

Value in $:	0	100	235.56

Exercise 2. Modify program so that it also converts the input value from dollars to pesos.

Demo your modified program for credit.