Object Oriented Software Development

Functions for All Subtasks

**Synopsis**

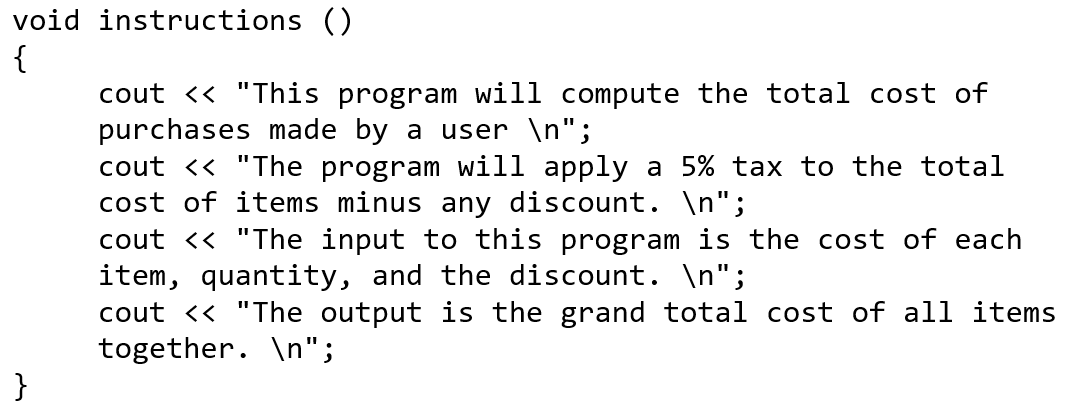
This laboratory session is intended enable you to:

* Learn about void functions (functions that do not return anything)
* Learn about functions calling other functions
* Learn about functions which return more than one value (*call\_by\_reference*)
* Learn about testing and debugging (stubs and drivers)

**void Functions**

So far, we have seen and used functions that return a value. Sometimes, it is convenient to create a function that carries out some activities but does not return a value to the calling function. Such a function may perform some calculations and display the result.  A good example of such a function is one that displays a set of instructions on the screen regarding how a program works.  Including the instruction function will improve the program.  If we put the extra information in the main function, we will clutter up the main and make it longer and more tedious to read. Instead, we can put the user instructions in a separate function named (what else?) "instructions". This function has a name (instructions), it does not have any arguments, it is of type void, i.e., it does not return any value.

Here is an example for the instruction function:



Note that since the function is of type void, no return statement is used.  However, a void function may have a return statement, but it will return nothing, i.e.,

return;

may be used in a void function as well.  The inclusion of return; in a void function may be inevitable when one wants to exit the function upon reaching a specific condition.  For example:

if( cost\_per\_item <=0)   
    return;   
else   
   do something  .....

Another example of a void function is a function that takes a real value and displays it with a specific number of decimal points.

void display\_it(double x, int precision)   
{   
      cout.setf(ios::fixed);   
      cout.setf(ios::showpoint);   
      cout.precision(precision);   
      cout << x << endl;   
}

Can you use overloading to write a function that displays two real values with a specific precision?

**Functions that “Return” More than One Value (Call-by-Reference)**

**HINT:** *"A function cannot return more than one value using the return statement, but can use the call-by-reference to update many values"*

In the previous lab, we saw several examples of functions that returned one value.  Sometimes we may want to return more than one value from a function.  The return statement can be used to return one value only.   Instead, we will use the **call-by-reference** mechanism to update the arguments that are passed to a function.  Following is an example in which we have used this method to return a value to the main function without using the return statement.  A call-by-reference parameter is marked by an **&** so that the compiler will distinguish it from other parameters. Suppose, you wanted to call a function that would ask users to input the coefficients, **a, b,** and **c**.  You could have called a function as:

get\_a\_b\_c(a,b,c);

Now, the declaration of this function will look like this:

void get\_a\_b\_c(float**&** a, float**&** b, float**&** c);

Let's look at a program and see how this works.  In the following example, the get\_input function obtains two values from the user, then returns (brings) them to the main function. In a sense, get\_input "returns" two values.  This cannot be done with the return statement because the return statement returns exactly one value. If a function must produce more than one output value, then we must use call-by-reference parameters (one for each output value).

**Questions**

1. Locate c5e1-1 and type the program into an editor. Save this as c5e1.cpp then compile and run.
2. Which one of the values, i or j, got updated upon return from the function process?
3. What would you do to get both values updated upon your return to the main? Make the necessary changes and run the program to make sure it works correctly.

Using & we access the address of a variable, thus, when we make a change in the value in a function, the change will be seen in the calling function as well, i.e., the   
value gets updated.

It is worth noting that one may think that the use of & in all function calls sounds like a good idea.  The argument one may make is that "it is safe to use it all the time, so when the value needs to get updated, it is already there."  However, passing a variable using call-by-reference may result in unwanted and undesirable changes in the value, which may cause the program to produce incorrect results.

As a rule, you should keep all parameters as call-by-value and only change those to call-by-reference when they need to be updated.

1. Create a new program called c5e2.cpp. The program should ask the user for two integer values and store them in *x* and *y*. The program will output the values in x and y. Next the program will call a function called swapXY and pass x and y by reference. The program will output the new values in x and y.

**Function Calling Another Function**

You have already seen a function that has called another function, but you may not have paid close attention to it.  In the previous labs, you called a predefined function from the main.  You also called the function get\_input from the main function in the previous lab.  In order to call a function inside another function, you need to have the declaration of the function that is being called before the declaration of the calling function.

void that\_function(double& x );   
double this\_function( ); //This function can now call that function

int main( )   
{   
  double y;   
  y = this\_function( );  
 return 0;   
}

double this\_function( )   
{   
      double x;   
      that\_function(x );  // that function is being called inside   
 // this function

      return x;   
}

void that\_function(double& x )   
{   
    ....   
}

The following function will use **a, b,** and **c** as the coefficients of a quadratic equation to compute **b**2 - 4**ac.**   This function calls on another function called get\_a\_b\_c to get the values for **a**, **b**, and **c.**

double bb\_4ac( )   
{   
 double a, b, c;  // Coefficients of a quadratic equation   
  get\_a\_b\_c(a, b, c);

  return b\*b - 4\*a\*c;   
}

**Questions**

1. Create a new program called c5e3.cpp. Write the full program that makes use of the bb\_4ac function.

**A Program with Several Functions, Testing and Debugging Functions-Stubs and Drivers**

In this part, we want to create a program that draws a diamond or a triangle with a size that the user selects using a specific character that will be entered at the keyboard.  Here are two examples:

A triangle of size 4, using \*:

\*   
      \*\*\*   
    \*\*\*\*\*   
  \*\*\*\*\*\*\*

A diamond of size 4, using \*:

\*   
      \*\*\*   
    \*\*\*\*\*   
  \*\*\*\*\*\*\*   
    \*\*\*\*\*   
      \*\*\*   
        \*

What do we need for this program?

**Problem definition**

We first need to clearly define the problem, the inputs and the outputs.

**Inputs:**

1) a choice to draw one of the two shapes or to quit the program,   
    2) a character choice which will be used for drawing the selected shape, and   
    3) the size of the shape

**Output:**

One of these will be the output:

1) a triangle,   
   2) a diamond,   
   3) quit

**Analysis of the problem**

We need to break down the tasks that this program is supposed to complete. The main tasks are:

1) ask for the input data   
    2) for both triangle and diamond, first display a triangle of requested size, and then if the

choice is a diamond, add the bottom part   
    3) displays the shape

To write this program, we will assume that we have the following functions:

1) *void instructions( )*   
*// This function describes the program and how it works*

2) *int menu( )*   
*// This function will return a choice to the main ; 1) draw triangle, 2) draw diamond, and*

*3) quit*

    3) *draw\_shape(int choice)*   
*// This function calls on appropriate function depending on the choice to draw a shape*

    4) *int get\_size( )*   
*// This function will return the size of the shape, same function for either of the shapes*

    5) *char get\_char( )*   
*// This function will ask users to select a character that will be used to draw the shape*

    6) *void draw\_triangle(int size, char c)*   
*// This function draws a  triangle of size size using character c*

*7*) *void draw\_diamond(int size, char c)*   
*// This function first calls draw\_triangle, then draw\_bottom to draw the diamond*

*8*) *void draw\_bottom(int size, char c)*   
*// This function actually draws an upside down triangle of size size-1 as the bottom of*

*the diamond*

Now that we have defined the functions, it is time to design the algorithm for the program.  However, we can write the main part of the program.

int main ( )   
{   
     int choice;

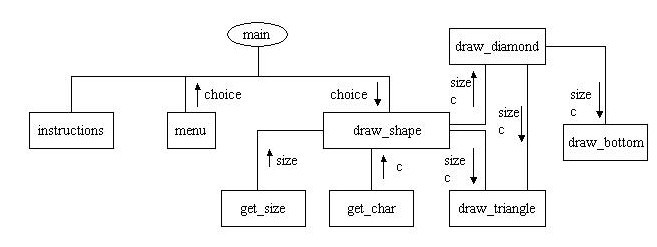
     instructions( );   
     choice = menu( );   
     if(choice != 1 || choice != 2)   
     {   
        cout << "You requested to quit, bye \n";   
        return 0;   
     }

     draw\_shape(choice);

     return 0;   
}

**Algorithm Design**

The following diagram (structure chart) summarizes the algorithm design for this program.  The direction of each arrow is the same as the direction in which the data flow.



**Testing and Debugging**

Each function should be designed, coded, and tested as a separate unit from the rest of the program.  To test a function, we will use a **driver program**.  It is impossible to test a function inside a function that has not been written.   In order to solve this problem, we use the simplified version of the function that is not written yet.  These functions are referred to as **stubs**.

Suppose we want to test the draw triangle function.  This function is called within the draw\_shape function.  There are two arguments to this function, the size and the character c. To test the program, we can use the driver shown below:

#include<iostream>   
using namespace std;

void draw\_triangle(int size, char c);   
void draw\_shape(int size, char c);

int main( )   
{   
// Simplified version of main, notice that char and size are fixed for now   
     char c = '\*';   
     int size = 4;

     draw\_shape(size, c);

     return 0;   
}

void draw\_shape(int size, char c) //A simplified version of draw\_shape   
{   
       draw\_triangle(size, c);   
}

void draw\_triangle(int size, char c)   
{   
  // complete version of draw\_triangle function   
  ....   
 ....   
}

**Questions**

1. Create a new program called c5e4.cpp. Continue with the process of creating the program to draw a triangle or a diamond based on the user’s choice, size and draw character.

**Submission Instructions**

1. Create a zipped folder with the following files:
   * c5e1.cpp
   * c5e2.cpp
   * c5e3.cpp
   * c5e4.cpp
2. Upload to the Moodle Link: Chapter 5 Lab Exercises