|  |  |  |
| --- | --- | --- |
| Starting Out With C++ |  | Early Objects 9th ed. |
|  | Lab 3 |  |
|  |  | Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**To begin**

* Log on to your system and create a folder named Lab3 in your work space.
* Copy all the files from Blackboard Lab 3 into yourLab3 folder.

**Turn in:**

**On Blackboard:** File areas4.cpp. At the end of the file paste sample output from the program. Turn in only one file.

**LAB 3.1 – TRY IT: Working with Functions and Function Calls**

**Step 1:** Add the tryIt6A.cpp program in your Lab3 folder to the project. Below is a partial listing of the source code.

|  |  |
| --- | --- |
| 1 // Lab 3 tryIt6A    12 /\*\*\*\*\* main \*\*\*\*\*/ 13 int main() 14 { int value = 2; 15  16 cout << "Hello from main.\n"; 17 printMessage(); 18  19 cout << "\nValue returned by tripleIt is " 20 << tripleIt(value) << endl; 21 cout << "In main value now is "  22 << value << endl << endl; 23  24 value = tripleIt(value); 25 cout << "In main value now is "  26 << value << endl; 27  28 value = tripleIt(value); 29 cout << "In main value now is "  30 << value << endl << endl; 31  32 cout << "Goodbye from main.\n"; 33 return 0; 34 } 35  36 /\*\*\*\*\* printMessage \*\*\*\*\*/ 37 void printMessage() 38 { 39 cout << "Hello from PrintMessage.\n"; 40 } 41  42 /\*\*\*\*\* tripleIt \*\*\*\*\*/ 43 int tripleIt(int someNum) 44 { 45 return someNum \* someNum \* someNum; 46 } | Expected Output |
| Observed Output |

**Step 2:** Read the source code, paying special attention to the flow of control from main to the functions it calls and then back to main again. Notice what main passes to each function and what, if anything, the function return. Once you have done this, complete the “Expected Output” box in the table above, writing down what the program will display in the order it will be displayed.

**Step 3:** Now compile and run the tryIt6A.cpp program, and look at the output it creates. If the actual output matches what you wrote down, just place a checkmark in the “Observed Output” box. If it is not the same, write down the actual output.

**LAB 3.2 – Using a void Function**

**Step 1:**  Open fortunes.cpp program. Below is a copy of the source code.

1 // Lab 3 fortunes.cpp   
 2 // This fortune telling program will be modified to use a void function.  
 3 // PUT YOUR NAME HERE.  
 4 #include <iostream>  
 5 #include <cmath>  
 6 using namespace std;  
 7   
 8 // Function prototype  
 9 // WRITE A PROTOTYPE FOR THE tellFortune FUNCTION HERE.  
10   
11 /\*\*\*\*\* main \*\*\*\*\*/  
12 int main()  
13 {  
14 int numYears,  
15 numChildren;  
16   
17 cout << "This program can tell your future. \n"  
18 << "Enter two integers separated by a space: ";  
19   
20 cin >> numYears >> numChildren;  
21   
22 numYears = abs(numYears) % 5; // Convert to a positive integer 0 to 4  
23 numChildren = abs(numChildren) % 6; // Convert to a positive integer 0 to 5  
24   
25 cout << "\nYou will be married in " << numYears << " years "  
26 << "and will have " << numChildren << " children.\n";  
27   
28 return 0;  
29 }  
30   
31 /\*\*\*\*\* tellFortune \*\*\*\*\*/  
32 // WRITE THE tellFortune FUNCTION HEADER HERE.

33 // WRITE THE BODY OF THE tellFortune FUNCTION HERE.

**Step 2:** Run the program to see how it works.What output do you get when you input the following values at the prompt? -99 14

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Step 3:** Create a function that contains thefortune telling part of the code by doing the following:

* On line 9 write the prototype for a void function named tellFortune that has two integer parameters.
* On line 32 write the function header for the tellFortune function. Following that should be the body of the function. Move lines 22 – 26 of the program to the function body.
* Replace current lines 22 – 26 of main with a call to the tellFortune function that passes it two arguments, numYears and numChildren.

**Step 4:** Recompile and rerun the program. Enter -99 and 14 again. It should work the same as before.

**Step 5:** If your professor asks you to do so, print the revised source code and the output of executing it several times, using a variety of inputs.

**LAB 3.3 – Modularizing a Program with void Functions**

**Step 1:** Open areas.cpp program in your Lab3 folder to the project. This file contains just a program shell in which you will write all the programming statements needed to complete the program described below. Here is a copy of the current contents of areas.cpp.

1 // Lab 3 areas.cpp   
 2 // WRITE A COMMENT BRIEFLY DESCRIBING THE PROGRAM.   
 3 // PUT YOUR NAME HERE.  
 4 // INCLUDE ANY NEEDED FILES HERE.  
 5 using namespace std;  
 6   
 7 int main()  
 8 {   
 9 // DEFINE THE NAMED CONSTANT PI HERE AND SET ITS VALUE TO 3.14159  
10   
11 // DECLARE ALL NEEDED VARIABLES HERE. GIVE EACH ONE A DESCRIPTIVE  
12 // NAME AND AN APPROPRIATE DATA TYPE.   
13   
14 // WRITE STATEMENTS HERE TO DISPLAY THE 4 MENU CHOICES.  
15   
16 // WRITE A STATEMENT HERE TO INPUT THE USER'S MENU CHOICE.  
17   
18 // USE AN IF/ELSE IF STATEMENT TO OBTAIN ANY NEEDED INPUT INFORMATION   
19 // AND COMPUTE AND DISPLAY THE AREA FOR EACH VALID MENU CHOICE.  
20 // IF AN INVALID MENU CHOICE WAS ENTERED, AN ERROR MESSAGE SHOULD  
21 // BE DISPLAYED.  
22   
23 return 0;  
24 }

**Step 2:** Design and implement the areas.cpp program so that it correctly meets the program specifications given below.

|  |  |
| --- | --- |
| ***Specifications:***  Create a menu-driven program that finds and displays areas of 3 different objects.  The menu should have the following 4 choices:  1 -- square  2 -- circle  3 -- right triangle  4 -- quit   * If the user selects choice 1, the program should find the area of a square. * If the user selects choice 2, the program should find the area of a circle. * If the user selects choice 3, the program should find the area of a right triangle. * If the user selects choice 4, the program should quit without doing anything. * If the user selects anything else (i.e., an invalid choice) an appropriate error message should be printed. | ***Sample Run***  Program to calculate areas of objects  1 -- square  2 -- circle  3 -- right triangle  4 -- quit  **2**  Radius of the circle: **3.0**  Area = 28.2743 |

**Step 3:** Once you have your program working, run it 4 times, each time testing a different valid menu choice. Then run it a fifth time using an invalid menu choice (such as 0 or 5). Make sure your program works correctly for all cases.

**Step 4:** Save the file areas.cpp as areas2.cpp

**Step 5:** Add a do-while loop to the program so that the user can repeatedly display the menu, make a choice, and have the appropriate steps for that choice carried out. The loop should continue iterating until the user enters 4 for the menu choice. Have the program print several blank lines after each case is carried out before the menu displays again.

**Step 6:** Once the program compiles with no errors, test it with a single run that tries all menu choices before entering 4 to quit.

**Step 7:** Save the file areas2.cpp as areas3.cpp

**Step 8:** Modularize the program by adding the following 4 functions. None of them have any parameters.

* void displayMenu()
* void findSquareArea()
* void findCircleArea()
* void findTriangleArea()

To do that you will need to carry out the following steps:

* Write prototypes for the four functions and place them above main.
* Write function definitions (consisting of a function header and initially empty body) for the four functions and place them below main.
* Move the appropriate code out of main and into the body of each function.
* Move variable definitions in main for variables no longer in main to whatever functions now using those variables. They will be *local* variables in those functions. For example, findSquareArea will need to define the side variable and findCircleArea will need to define the radius variable. All of the functions that compute areas will now need to define a variable named area.
* Move the definition for the named constant PI out of main and place it above the main function.
* In main, replace each block of removed code with a function call to the function now containing that block of code.

**Step 9:** Compile the code, fixing any errors until it compiles without errors. Then test it. Make sure it runs correctly for all menu choices. **LAB 3.4 – Using a Function that Returns a Value**

**Step 1:** Open choice.cpp program in your Lab3 folder to the project. Below is a copy of the source code.

1 // Lab 3 choice.cpp   
 2 // This program illustrates how to use a value-returning   
 3 // function to get, validate, and return input data.  
 4 // PUT YOUR NAME HERE.  
 5 #include <iostream>  
 6 #include <cmath>  
 7 using namespace std;  
 8   
 9 // Function prototype  
10 int getChoice();  
11   
12 /\*\*\*\*\* main \*\*\*\*\*/  
13 int main()  
14 {  
15 int choice;  
16   
17 cout << "Enter an integer between 1 and 4: ";  
18   
19 // WRITE A LINE OF CODE TO CALL THE getChoice FUNCTION AND TO  
20 // ASSIGN THE VALUE IT RETURNS TO THE choice VARIABLE.  
21   
22 cout << "You entered " << choice << endl;  
23 }  
24   
25 /\*\*\*\*\* getChoice \*\*\*\*\*/  
26 int getChoice()  
27 {  
28 int input;  
29   
30 // Get and validate the input  
31 cin >> input;  
32 while (input < 1 || input > 4)  
33 { cout << "Invalid input. Enter an integer between 1 and 4: ";  
34 cin >> input;  
35 }  
36 return input;  
37 }

**Step 2:** Read through the code to see how it works. Notice that the getChoice function validates the input before returning it.

**Step 3:** Follow the directions given in the uppercase comments on lines 4 and 19-20. Then compile and run the program. When prompted for an input, use the data shown in the sample run below. You should get the same results.

*Sample Run*

Enter an integer between 1 and 4: **0**

Invalid input. Enter an integer between 1 and 4: **9**

Invalid input. Enter an integer between 1 and 4: **2**

You entered 2

**Step 4:** Now make the getChoice function more versatile so it can validate that a choice is in any desired range, not just 1 – 4. Do this by carrying out the following steps:

* Add two integer parameters named min and max to the function header and modify the function prototype to agree with this.
* Revise the function so that it now validates that the input is between min and max. Remember to change the error prompt as well as the test condition of the while loop.
* Revise the line of code in main that calls the function so that it now passes two arguments to the function. Pass the values 1 and 4 to the function (though other values would work also).

Now recompile and rerun the program, again using the data from the sample run shown above. The program should produce the same results as it did before.

**Lab 3.5 – Modularizing a Program with Value-Returning Functions**

**Step 1:** In your Lab3 folder make a copy of your areas3.cpp file. Name it areas4.cpp

**Step 2:** Open areas4.cpp program to the project.

**Step 3:** Copy the getChoice function you just wrote in the choice.cpp file for the Lab 6.4 exercise and paste it below the displayMenu function definition in the areas4.cpp file. Add a function prototype for the getChoice function at the top of the program where the other prototypes are located. Now, change the following line of code in main

cin >> choice;

to

choice = getChoice(1, 4);

This will ensure that choice is assigned a value between 1 and 4. Therefore the final else if can be removed from the if/else if statement that controls the branching. After doing this, test the program to make sure everything works so far, before going on to the next step.

**Step 4:** Now, make the findSquareArea, findCircleArea, and findTriangleArea functions into value-returning functions. They should each return a double value. Change their function headers and function prototypes to indicate this. Then, instead of having them *print* the area, have them *return* the area they have computed. Finally, change the call to each of these functions in main so that the value returned by the function call will be printed. For example, you will change

if (choice == 1)

findSquareArea();

to

if (choice == 1)

cout << "Area = " << findSquareArea() << endl;

**Step 5:** Compile the code, fixing any errors until it compiles without errors. Then test it. Make sure it runs correctly for all menu choices.

**Lab 3.6 – Using Value and Reference Parameters**

**Step 1:**  Open swapNums.cpp program in your Lab3 folder to the project. Below is a copy of the source code.

1 // Lab 3 swapNums.cpp -- Using Value and Reference Parameters  
 2 // This program uses a function to swap the values in two variables.  
 3 // PUT YOUR NAME HERE.  
 4 #include <iostream>  
 5 using namespace std;  
 6   
 7 // Function prototype  
 8 void swapNums(int, int);  
 9   
10 /\*\*\*\*\* main \*\*\*\*\*/  
11 int main()  
12 {  
13 int num1 = 5,  
14 num2 = 7;  
15   
16 // Print the two variable values  
17 cout << "In main the two numbers are "   
18 << num1 << " and " << num2 << endl;  
19   
20 // Call a function to swap the values stored   
21 // in the two variables  
22 swapNums(num1, num2);  
23   
24 // Print the same two variable values again  
25 cout << "Back in main again the two numbers are "   
26 << num1 << " and " << num2 << endl;  
27   
28 return 0;  
29 }  
30   
31 /\*\*\*\*\* swapNums \*\*\*\*\*/  
32 void swapNums(int a, int b)   
33 { // Parameter a receives num1 and parameter b receives num2  
34 // Swap the values that came into parameters a and b  
35 int temp = a;  
36 a = b;  
37 b = temp;  
38   
39 // Print the swapped values  
40 cout << "In swapNums, after swapping, the two numbers are "   
41 << a << " and " << b << endl;  
42 }

**Step 2:** Read the source code, paying special attention to the swapNums parameters. When the program is run do you think it will correctly swap the two numbers? Compile and run the program to find out.

Explain what happened. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Step 3:** Change the two swapNums parameters to be reference variables. Section 6.13 of your text shows how to do this. You will need to make the change on both the function header and the function prototype. Nothing will need to change in the function call. After making this change, recompile and rerun the program. If you have done this correctly, you should get the following output.

In main the two numbers are 5 and 7

In swapNums, after swapping, the two numbers are 7 and 5

Back in main again the two numbers are 7 and 5

Explain what happened this time. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Lab 3.7 – Complete Program**

**Step 1:** Open the kiloConverter.cpp program in your Lab6 folder to the project. This file contains just a program shell in which you will write the programming statements needed to complete the program described below. Here is a copy of the file.

1 // Lab 3 kiloConverter.cpp   
 2 // This menu-driven program lets the user convert   
 3 // pounds to kilograms and kilograms to pounds.  
 4 // PUT YOUR NAME HERE.  
 5 #include <iostream>  
 6 using namespace std;  
 7   
 8 // Function prototypes  
 9 // WRITE PROTOTYPES FOR THE displayMenu, getChoice,  
10 // kilosToPounds and poundsToKilos FUNCTIONS HERE.  
11   
12 /\*\*\*\*\* main \*\*\*\*\*/  
13 int main()  
14 {  
15 // DECLARE ANY VARIABLES MAIN USES HERE.  
16   
17 // WRITE THE CODE HERE TO CARRY OUT THE STEPS  
18 // REQUIRED BY THE PROGRAM SPECIFICATIONS.  
19   
20 return 0;  
21 }  
22   
23 /\*\*\*\*\* displayMenu \*\*\*\*\*/  
24 // WRITE THE displayMenu FUNCTION HERE.  
25 // THIS void FUNCTION DISPLAYS THE MENU CHOICES  
26 // 1. Convert kilograms to pounds  
27 // 2. Convert pounds to kilograms  
28 // 3. Quit  
29   
30 /\*\*\*\*\* getChoice \*\*\*\*\*/  
31 // THIS IS THE SAME FUNCTION YOU WROTE EARLIER IN THIS SET  
32 // OF LAB EXERCISES. JUST FIND IT AND PASTE IT HERE.   
33   
34 /\*\*\*\*\* kilosToPounds \*\*\*\*\*/  
35 // WRITE THE kilosToPounds FUNCTION HERE.  
36 // IT RECEIVES A WEIGHT IN KILOS AND MUST CALCULATE  
37 // AND RETURN THE EQUIVALENT NUMBER OF POUNDS.  
38   
39 /\*\*\*\*\* poundsToKilos \*\*\*\*\*/  
40 // WRITE THE poundsToKilos FUNCTION HERE.  
41 // IT RECEIVES A WEIGHT IN POUNDS AND MUST CALCULATE  
42 // AND RETURN THE EQUIVALENT NUMBER OF KILOS.

**Step 2:** Design and implement a modular, menu-driven program that converts kilograms to pounds and pounds to kilograms. 1 kilogram = 2.2 pounds. The program should display a menu, accept and validate a user menu choice, get the amount of weight to be converted, call the appropriate function to do the conversion, and then print the returned result. The code should continue iterating to allow additional conversions to be done until the user enters the menu choice to quit. When the program runs it should look somewhat like the sample run shown here.

*Sample Run*

1. Convert kilograms to pounds

2. Convert pounds to kilograms

3. Quit

**1**

Weight to be converted: **4**

4 kilograms = 8.8 pounds.

1. Convert kilograms to pounds

2. Convert pounds to kilograms

3. Quit

**2**

Weight to be converted: **10**

10 pounds = 4.54545 kilograms.

1. Convert kilograms to pounds

2. Convert pounds to kilograms

3. Quit

**3**

**Step 3:** Once your program is written and compiles with no errors, thoroughly test it.