|  |  |
| --- | --- |
|  | **CS 1400 Lab #20**  **Passing by Value, Passing by Reference and passing by out**  **Version 1.0**  **Introduction**  In this exercise, you will create a method that passes one or more of its parameters by **value**, by **reference** and by **out**.  **The Problem**  In this exercise, you will write a program that will determine how many coins of each denomination (1¢, 5¢, 10¢, 25¢, and 50¢) are required to make up a given amount of change. For example, to make 57¢ we could use 1 half dollar, 1 nickel, and 2 pennies.  **Deriving the Solution**  Given an amount of change to make, it should be obvious that many combinations of coins are possible. In the example above, we could make up 57¢ with: 57 pennies; 5 dimes, a nickel, and 2 pennies; 2 quarters and 7 pennies; and so on. For this exercise let us use an algorithm that figures out how many of the largest denomination coin we need first, and works down through each succeeding coin denomination. This algorithm is described in the following diagram.  making change  To see how the algorithm works, let's take our 57¢. First consider how many 50¢ pieces we can get out of 57¢, we can get 1. After taking out the 50¢, we have 7¢ left. Now consider quarters. We can't get any quarters out of 7¢. Then look at dimes. We can't get any dimes out of 7¢ either. Then look at nickels. We can take one nickel out of 7¢. This leaves 2¢, and finally we have two pennies.  **ComputeChange( ) Method**  If you look carefully at the Activity Diagram above, you will notice that the code does the same thing over and over again, but with different values for n and for the coin denomination. Whenever you see this pattern in a program, you should consider writing a method. In this case, you want a method that takes as its inputs n, the amount of change left, and a coin denomination. The method should compute the number of coins of the given denomination it can take out of n, and compute the new value of n, after taking out those coins.  We now have a problem. We need to return both the number of coins we computed, **and** the new value of n. However, the method is restricted to void as it’s return type. Also we need to return both amount of money left after taking the coins out and the count of these coins. The solution here is to pass the value of n by reference and the number of coins by out. When the new value of n is computed, it is computed in place, so we do not need to return the value. We only need to return the computed number of coins. The signature for the method would then look like  ***static void ComputeChange(ref int changeValue, int coinValue, out int nCoins);***  **Writing the Code**  Use the .cs file that is located below.  **// Compute change program**  **// Use this code**  **using** **System**;  **static** **class** Program  {  **// some class level constants**  **enum** ECOINS {**HALVES** **=** 50, **QUARTERS** **=** 25, **DIMES** **=** 10, **NICKELS** **=** 5, **PENNIES** **=** 1 }  **static** **int**[] **COINS** **=** {(**int**)ECOINS**.HALVES**, (**int**)ECOINS**.QUARTERS**,  (**int**)ECOINS**.DIMES**, (**int**)ECOINS**.NICKELS**,  (**int**)ECOINS**.PENNIES** };  **const** **string** **MSG\_HEADER** **=** **"I will make change for you."**;  **const** **string** **MSG\_ENTER** **=** **"Enter in an amount between 1 and 99: "**;  **const** **string** **MSG\_GBYE** **=** **"Goodbye ... "**;  **const** **string** **MSG\_INVALID** **=** **"Invalid int value for money!"**;  **const** **string** **MSG\_MNY** **=** **"For your money {0} cents you get:"**;  **const** **string** **MSG\_COIN** **=** **"{0} halves"**;  **static** **string**[] **SCOINS** **=** {**"Halves"**,**"Quarters"**,**"Dimes"**,**"Nickels"**,**"Pennies"**};  **static** **void** **Main**()  {  **CalcChange**();  }**//End Main()**  **// The CalcChange Method**  **// Add your method prolog here**  **public** **static** **void** **CalcChange**()  {  **int** **money** **=** 0; **// the value we want to count change for**  **int** **numberCoins** **=** 0;  **string** **inputStg** **=** **null**;  **do**  {  Console**.Clear**();  Console**.WriteLine**(**MSG\_HEADER**);  Console**.Write**(**MSG\_ENTER**);  **if** (**int.TryParse**((**inputStg** **=** Console**.ReadLine**()), **out** **money**) **==** **false**)  {  **if** (**inputStg** **==** **""**)  {  Console**.WriteLine**(**MSG\_GBYE**);  Console**.ReadLine**();  **return**;  }  Console**.WriteLine**(**MSG\_INVALID**);  Console**.ReadLine**();  **continue**;  }  Console**.WriteLine**(**MSG\_MNY**, **money**);  **for** (**int** **index** **=** 0; **index** **<** **COINS.Length**; **index++**)  {  **ComputeChange**(**ref** **money**, **COINS**[**index**], **out** **numberCoins**);  Console**.WriteLine**(**MSG\_COIN** **+** **SCOINS**[**index**], **numberCoins**);  }  Console**.ReadLine**();  } **while** (**true**);  }**//End CalcChange( )**  **// The ComputeChange Method**  **// Add your method prolog here**  **static** **void** **ComputeChange**(**ref** **int** **changeValue**, **int** **coinValue**, **out** **int** **nCoins**)  {  **// you provide the method to compute change here!**  }**//End ComputeChange( )**  }**//End class Program**  Where shown in the code, write the ComputeChange method, following the steps in the Activity Diagram above. Write a method Prolog for your method. Before compiling your program do a code walkthrough as described in the reading material for this lab. Test and debug your program.  **File(s) to Submit:**  Place the complete project folder in a zip file and name the zip file Lab\_20\_your-initials\_V1.0.zip. For example, I would name my file Lab\_20\_DAF\_V1.0.zip. Submit this assignment as Lab #20 on Canvas. |

|  |  |  |
| --- | --- | --- |
|  | ***Grading Checklist*** |  |
| # | Program | C(correct)  X(incorrect) |
| 1 | Meets & works to specifications | 6 points |
| 2 | Error Free, elegant & efficient | 4 points |
| 3 | Pseudo-Code | -3 points |
| 4 | Style Guidelines | -2 points |
| 6 | Source Files(s) & Formatting | -2 points |
| 7 | Project Prolog | -1 points |
| 8 | Function Prologs | -1 points |
| 9 | Zip Filename | -1 points |
| 10 | Lab & Project Names | -1 points |
| 11 | Zip File is invalid or will not unzip | Lab = 0 pts |
|  | Total Points | 10 | 0-9 |