Change Maker in C ++

In C ++, the different implementations of the change maker program have different approaches. Working with the “int” data type requires far less complexity than that of “float” or “double” due to the cut-off nature of the int data type. Since float and double both incorporate decimal values, this requires the programmer to provide code that is more restrictive to prevent errors in the results. Because of this, the int option is more clean and simplified, leading to greater readability, writability, and reliability.

Out of the three implementations of change maker, due to the restrictive nature of the int data type, this version provides the most readability since the code is strait-forward and is easy to follow. One problem with using int, which is true with each of the three different forms, is that the readability leaves desired improvement. While the program’s output is simple, the code requires more complexity. This is because we need to convert out input into whole numbers to determine the output. Once this is finished, we must move through each level of currency.

With the second version, float, the data type allows for decimals. While this may appear to have greater writability, this actually creates more difficulties, as the programmer must first use the data given by the user, and ensure, at every step, that the output is a whole number. To do this, the programmer must determine whether ty need to round up, or down. Because of this, the programmer must also ensure that the output does not have a negative balance, while in the int data type, if the amount paid is greater than the cost of the program, this issue does not arise.

When working with double in C ++, the issues that arise are virtually identical with float, as the prim

ary difference between these two data types is that double provides greater accuracy. To be more precise, double data has a larger bit size, and thus holds more digits. In this instance, since only the first two decimals are relevant, both float and decimal produce the same result when going through the same process. Because of this similarity, the readability, and writability are the same. In regards to cost and reliability; however, decimal requires more data storage due to its size, but is more reliable as a whole.

Insertion Sort (re-visited) in C ++

In this program, the user is asked for the size of an array, and then for the numbers to fill the array. From here, the program will sort (using Insertion Sort) the array. The benefits of using an array in this program is that, since the size is finite, the program will not take longer to reference elements of the array, and can instead go to the specific array elements. The problem with this is that the user cannot add new elements to the array because the array is static.

Other types of lists, or collections of data, that would be useful in C ++ are vectors and linked lists. Vectors are far more common, due the fact that the data type is included as a library, whereas the user must define a linked list. The benefit of these two is that the size can be altered. With vectors, the user can add elements by “Pushing” them onto the end. A major benefit of this is that the vector has similar costs, while improved reliability and readability. With linked lists, the user can define each element; and the process of having them linked is by having one element “point” to the next. So to add new elements, the user simply must append the last element to point to the new element, or insert the element by altering the pointers.