# Number to Words

There are numerous approaches to solving the problem of converting numbers into words, some of which are discussed below.

## Place value if Statements

One alternative solutions is to create a series of if statements, each dealing with a different set of place values, such as hundreds, thousands and millions. This was my initial approach, however as the solution evolved it became clear that there was a lot of similar code being duplicated for each if statement. Even if this duplicate code was extracted into a separate function, the overhead of adding a new if statement for each new unit seemed too high.

if (Number > 999999)

{

result += RecursiveFunction(Number) + " million";

}

else if (Number > 999)

{

result += RecursiveFunction(Number) + " thousand";

}

else if (Number > 99)

{

result += RecursiveFunction(Number) + " hundred";

}

After partially implementing this solution, and thinking about the problem further, it was decided that if the relevant place value could be determined as part of the procedure then these additional if statements could be removed.

## Splitting into an array

Splitting the number into an array of either char or int values and iterating through the array and returning the respective word is another alternative. This solution was discarded primarily because it soon became clear that to return the correct word for any number between 10 and 99 required not only the current value, but also the next value.

Adding the place value to the resulting string could also prove difficult as you have to rely on the current position in the array to determine what place value should be applied.

char[] numbers = Number.ToString().ToCharArray();

for (int i = 0; i < numbers.Length; i++)

{

Converter(numbers[i]);

}

## Dictionary Values

Storing numbers and their relevant word in a dictionary, or list instead of an array was option that although valid, seemed to add little to the overall solution and was discarded.

Dictionary<int, string> wordsAndNumbers = new Dictionary<int, string>

{

{1, "one"},

{2, "two"},

{3, "three"},

{4, "four"},

{5, "five"}

};

// get value from dictionary

var wordValue = wordsAndNumbers[Number];

## Converting with units of measurement

A function that returned the result including the units, such as dollars and cents is possible, but was discarded on the grounds that the final presentation of the result should not be the responsibility of the main converter. It was determined that adding the units should be handled by the calling function, so as to make the converter more generic and reusable. This also meant that the converter only needed to deal with whole numbers as the calling function would be responsible for handling the splitting of decimals. Dealing in integers means that the converter could be used for other situations, such as retuning the words for coordinates written in degrees, minutes and seconds.