

Problem Solving by Computer – Logical Thinking

Tech Capsule 2 – Loop based problems

While designing an algorithm for a computational problem, there might be the need for doing the same task more than once. In such a case our first thought would be to write the required task N (number of times required) number of times. By doing so, we are able to get the required output/solution but it won't be considered as the best possible solution, as it would need the same lines of code again and again.

A better way would be to run a loop (which keeps running till a specific end condition is not reached) for N number of times required for the task.

Let us understand this more in detail with the help of an example.

Problem –

Design an algorithm that accepts a number and displays each digit of the number separately.

For example, if the input number is 254 the output should be printed as 2, 5, 4.

Solution –

We all know that a given number can also be represented as –

$$254 == 2 * 10^2 + 5 * 10^1 + 4 * 10^0$$

This would be our first level of thought. Now how do we get 2, 5, 4 separated and which operation must be used?

Let us look at an operator called the *modulus* operator or *mod* operator. Mod is an operator which gives us the remainder of a division operation on 2 numbers. (mod operators are implemented in all programming languages with different names).

Remainder step - $254 \text{ mod } 10$ gives us 4.

So by this we get the right most digit i.e. 4

Now next step is to extract the digit 5 and to do so we can now only need 25 from 254

Division step - Let us use another operator called the division operator on the number 254.

$254 \text{ divide } 10$ gives us 25

Now follow it up with the remainder operator for 25

Remainder step - $25 \text{ mod } 10$ gives us 5

So with this we have extracted the digits 4 and 5.

Continue in this way,

Division step - 25 **divide** 10 gives us 2

Remainder step - 2 **mod** 10 gives us 2

Thus getting the 3 digits 4, 5, and 2 which can be displayed as mentioned in the question.

As we understand the approach, we can observe that the divide by 10 and remainder operator 10 task is repeated 3 times as there are 3 digits in the number. If there were n digits in the number it would have to be repeated n number of times. So to write an algorithm which works irrespective of the change in number of digits we must look at a looping mechanism which repeats the tasks the number of times required based on the number of digits present.

Let us call that looping mechanism as a **while** loop.

Note – Every programming has various looping mechanisms for example a **for** loop, a **while** loop, a **do-while** loop. Here in our example we will consider a while loop.

The algorithm for the problem would now look as below.

Step 1: Read a number N from the user

Step 2:

while integer being reversed is greater than 0

- A- Do $N \bmod 10$ to extract the last digit. Save it in variable X
- B- Display the variable X
- C- Do $N \text{ divide by } 10$ to get the new number for next iteration.
- D- Repeat

Step 3: End

Try by yourself

Activity 1:

Design an algorithm to count the number of digits in a given number.

Example –

If the number entered is 12345

The output is 5 digits

In the number entered is 123

The output is 3 digits

Activity 2:

Design an algorithm to find the sum of all the digits in a given number.

Example –

If the number entered is 12345

The output is 15

Activity 3:

Design an algorithm which accepts a decimal integer and then displays its corresponding binary representation.

Activity 4:

Design an algorithm which accepts a binary representation of a number and then displays its corresponding decimal equivalent.

Activity 5:

Design an algorithm which accepts a number from the user and displays its smallest exact divisor other than one.