Let S be a 0-indexed string of n digit characters and let S_i denote the i^{th} character in S. The integer value of S, call it a, is given by

$$a = 10^{n-1}S_0 + 10^{n-2}S_1 + \cdots + 10S_{n-2} + S_{n-1}$$

Or, equivalently,

$$a = S_{n-1} + 10S_{n-2} + \cdots + 10^{n-2}S_1 + 10^{n-1}S_0$$

Define the function C which takes a character digit and converts it to an integer. Then the function F can be recursively defined to convert a string to an integer:

$$F(S) = \begin{cases} C(S_0) & n = 1 \\ C(S_{n-1}) + 10F(S_{[0:n-1]}) & n > 1 \end{cases}$$

where $S_{[a:b]}$ denotes the substring of S from index a to index $b - 1$.

We demonstrate this function on the string "12345":

$$\begin{aligned} F(\text{“12345”}) &= C(\text{‘5’}) + 10F(\text{“1234”}) \\ &= 5 + 10(C(\text{‘4’}) + F(\text{“123”})) \\ &= 5 + 10(4 + 10(C(\text{‘3’}) + 10F(\text{“12”})) \\ &= 5 + 10(4 + 10(3 + 10(C(\text{‘2’}) + 10F(\text{“1”}))) \\ &= 5 + 10 \cdot 4 + 100(3 + 10(2 + 10C(\text{‘1’}))) \\ &= 5 + 10 \cdot 4 + 100 \cdot 3 + 1000(2 + 10(1)) \\ &= 5 + 10 \cdot 4 + 100 \cdot 3 + 1000 \cdot 2 + 10000 \cdot 1 \\ &= 12345 \end{aligned}$$

In C++, we define the function C as **chtoi**:

```
int chtoi(char c)
{
  return c - '0';
}
```

This allows us to define the function F as **stoi** in the following way:

```
int stoi(string c, int n)
{
  if (n == 1)
  {
    return chtoi(c[0]);
  }
  else
  {
    return chtoi(c[0]) + 10 * stoi(c.substr(0, n - 1));
  }
}
```

We pass the length of the string, n, as a parameter of **stoi** in order to avoid recalculating the length of each substring.