**Number Of Digits In Factorial:**

**Number of Digit of an Integer**

First, we will apply the logarithm idea on an integer. So where is the connection of logarithm with digit number? Look at the following logarithm values:

log10(x)=y  
log10(1)=0

log10(10)=1

log10(100)=2

log10(1000)=3

log10(10000)=4

As the value of x increases, value of y also increases. Every time we multiple x by 10, value of y increases by 1. That is, every time number of digit increases, value of y increases. From this table, we can infer few things about log of other values.

If log10(100) is 2, and log10(1000) is 3, then for all value of x where 100<x<1000 , value of y will lie between 2<y<3. Let us try this out.

log10(100)=2

log10(150)=2.17609125906

log10(500)=2.69897000434

log10(999)=2.99956548823

Now note that, for every 100≤x<1000, value of y is 2≤y<3. Can you see some relation between value of y and number of digits of x?

Yes. If the value of y is of form 2.XXX, then x has 2+1=3 digits.

∴number of digits ofx=⌊log10(x)⌋+1

Be careful, ⌊log10(x)⌋+1 is not same as ⌊log10(x)+1⌋. Try it out with 100,1000,10000. We need to floor the log value **before**we add 1.

|  |  |
| --- | --- |
| 1  2  3  4  5 | int numberDigit ( int n ) {      int wrongAnswer = log10(n) + 1; ///This is wrong.      int rightAnswer = ( (int) log10(n) ) + 1; ///This is right.      return rightAnswer;  } |

In line 3, we type cast log10(n) to integer. This has same action as floor() function. Also note that we used log10() function instead of log() function. Unlike our calculators, in C++ log() has base 2.

**Extending To Factorial**

So how do we extend this idea to N!?

Let x=log10(N!). Then our answer will be res=⌊x⌋+1. So all we need to do is find value of x.

x=log10(N!)

x=log10(1×2×3×...×N)

∴x=log10(1)+log10(2)+log10(3)+...+log10(N) This is using the law log10(ab)=log10(a)+log10(b)

So in order to calculate x=log10(N!), we don't have to calculate value of N!. We can simply add log value of all numbers from 1 to N. This can be achieved in O(N).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | int factorialDigit ( int n ) {      double x = 0;      for ( int i = 1; i <= n; i++ ) {          x += log10 ( i );      }      int res = ( (int) x ) + 1;      return res;  } |

**Digits of**N!**in Different Base**

Now what if we want to find how many digits N! has if we convert N! to some other base.

For example, how many digits 3! has in binary number system with base 2? We know that (6)10=(110)2. So 3! has 3 digits in base 2 number system.

Can we use logarithms to solve this problem too? Yes.

number of digits of x in base B=logB(x)

All we need to do is change the base of our log and it will find number of digits in that base.

But, how do we change base in our code? We can only use log with base 2 and 10 in C++. Fear not, we can use the following law to change base of logartihm from B to C.

logB(x)=logC(x)logC(B)

So in C++, we will use C=2 or C=10 to find value of logB(x).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | int factorialDigitExtended ( int n, int base ) {      double x = 0;      for ( int i = 1; i <= n; i++ ) {          x += log10 ( i ) / log10(base); ///Base Conversion      }      int res = ( (int) x ) + 1;      return res;  } |