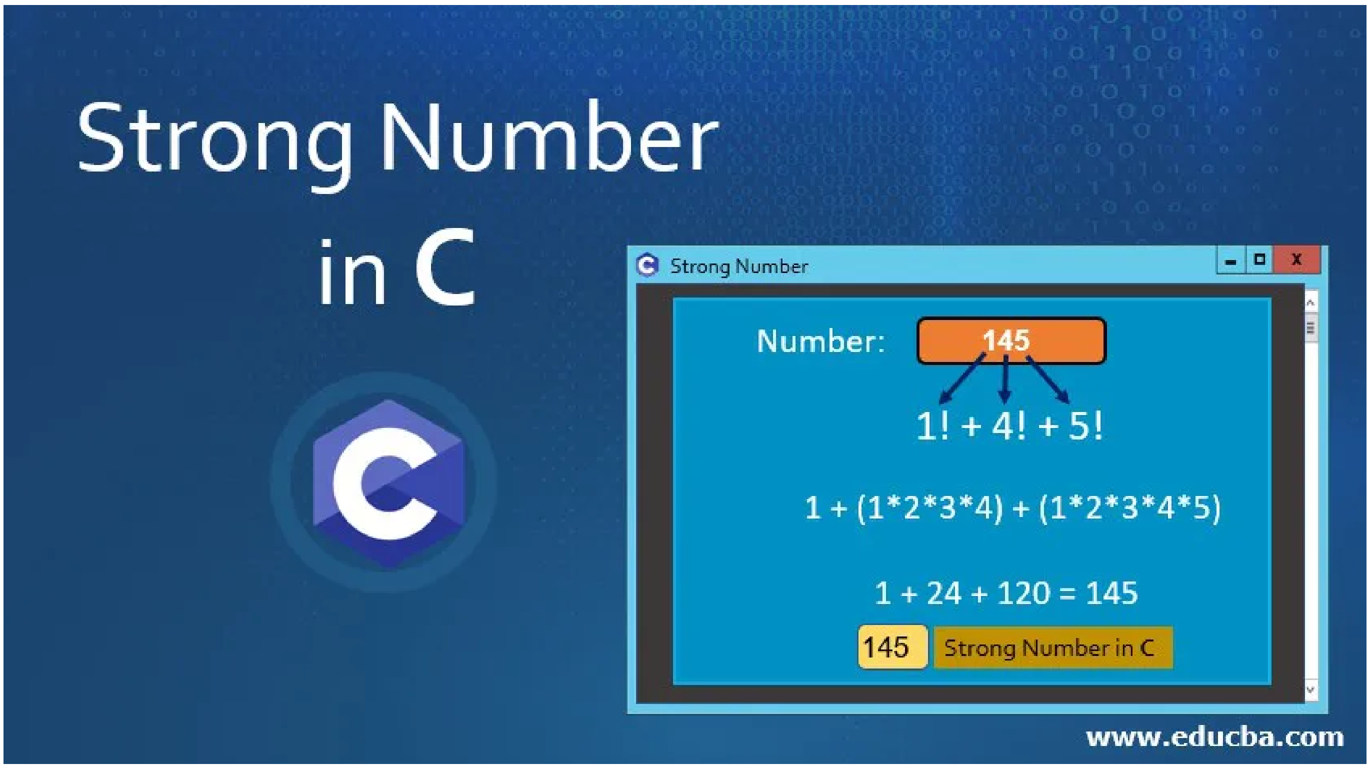
Strong Number in C

(https://www.educba.com/software-development/)

By Anusua Dutta (https://www.educba.com/author/anusua-dutta/)

← (https://www.educba.com/anagram-program-in-c/)

→ (https://www.educba.com/atm-program-in-c/)



Introduction to Strong Number in C

In C programming language, a number is said to be a strong number when the factorial individual number sums to the actual number. Strong Numbers are the numbers whose sum of the factorial of digits is equal to the original number.

Example:

- Take one input as 145.
- Find the sum of individual digits with its factorials 1! + 4! + 5! [Factorial plays an important role as output must compute the product of the number].
- The computed value should be the same as the original number.
- Output, in this case, is the same as 145 thus it results that it is a correct input as a strong number in C.

Logic Behind Strong Number in C

The logic behind defining and describing Strong Number remains same as explained in the introduction or definition of Strong Number in C. Say, someone, take a number in a manner that sum of individual number contributes to give a number which is the original number then it will not be considered as a Strong Number. That assumption and logic are totally wrong. Therefore, actual computation of Strong Number includes or considers factorial as a major logic flow beside it. Both the sum and factorial of individual number should define the Strong number.

There are some logical steps to be followed which are as follows:

1. Take input from a user to verify for a strong number. Assign a variable to say x to the number. Copy it to a temp variable for any further manipulation say it is assigned as y = x.

2. Consider and initialize another variable to store the sum of all individual digits as sum=0.

3. Search and get the last digit of the entire number y [temporary variable]. Assign the r a new variable, say z = y % 10.

4. Then, calculate the factorial of the last digit of the number. Store that factorial in a variable, say I which is a variable to store the result of the factorial of the numbers.

5. Add that final factorial to the variable of the sum which is initially 0 as sum = sum + i

6. Remove the last number which says x i.e. a temp variable.

7. Till y becomes more than 0 i.e. y>0 repeats all the steps from 3 to 6.

8. If sum becomes equal to the original or actual number after applying looping check c in the strong number. Then any given number is considered as Strong otherwise it is not a strong number.

QUIZ

9. Further examples will clarify the actual and crux logic behind the Strong Number. That misconception of adhering only to the individual number sum or product won't be enough. From this, a fact can be concluded that both sums of the individual number and product i.e. factorial of the individual digits contribute a lot for enhancing the entire logic behind strong Number in C.

How to Check Strong Number in C Using Various Methods?

Various Methods to check whether a number is a strong number or not are as follows:

- Using Functions
- Using for Loop
- Using While Loop

Example #1 – Using Functions

Code:

```
#include<stdio.h>
int fact (int val)
{
int a, b = 1;
for (a = 1; a <= val; a++)
{
b = b * a;
}
return b;
}

int main ()
{
int n, count, final_result, rem, sum = 0, tmp;
printf ("Enter a Number:\t");
scanf ("%d", &n);
tmp = n;
for (tmp = n; n > 0; n = n / 10)
{
count = 1, final_result = 1;
rem = n % 10;
final_result = fact(rem);
sum = sum + final_result;
}
if (sum == tmp)
{
printf ("%d is a Strong Integer\n\n", tmp);
}
else
{
printf ("%d is Not a Strong Integer\n\n", tmp);
}
return 0;
}
```

Output:

Example #2 – Using for Loop

Code:

```
#include <stdio.h>
```

```
#include<stdio.h>
int main ()
{
int n, cnt, fact, rem, sum = 0, tmp;
printf ("Enter a Number:\t");
scanf ("%d", &n);
for (tmp = n; n > 0; n = n / 10)
{
fact = 1;
rem = n % 10;
for (cnt = 1; cnt <= rem; cnt++)
{
fact = fact * cnt;
}
sum = sum + fact;
}
if (sum == tmp)
{
printf ("%d a Strong Integer \n\n", tmp);
}
else
{
printf ("%d Not a Strong Integer \n\n", tmp);
}
return 0;
}
```

Output:

Example #3 – Using While Loop

Code:

[(https://www.educba.com/software-development/)](https://www.educba.com/software-development/)

```
#include<stdio.h>
#include<conio.h>
int main ()
{
int n, cnt, fact, rem, sum = 0, tmp;
printf ("Enter a Number:\t");
scanf ("%d", &n);
tmp = n;
while(n)
{
cnt = 1, fact = 1;
rem = n % 10;
while (cnt <= rem)
{
fact = fact * cnt;
cnt++;
}
sum = sum + fact;
n = n / 10;
}
if (sum == tmp)
{

printf ("%d is a Strong Integer\n\n", tmp);
}
else
{
printf ("%d is Not a Strong Integer\n", temp);
}
return 0;
}
```

Output:

Example #4

To find a strong number between any given range.

Code:

```
#include<stdio.h>
#include<conio.h>
int main ()
{
int a, z, i, n, j, fact, sum;
printf ("Enter the range a and z Values (a<z): ");
scanf ("%d %d", &a, &z);
printf ("Strong numbers are:\n");
for (i=a; i<=z; i++)
{
n = i;
sum = 0;
while(n!=0)
{
fact=1;
for (j=1; j<=(n%10); j++)
{
fact *= j;
}
sum += fact;
n /= 10;
}
if(sum==i)
printf ("%d\n", i);
}
return 0;
}
```

Output:

Example #5

To find strong Numbers between 1 to n or below n.

Code:

```
#include <stdio.h>
#include <conio.h>
int main ()
{

int k, l, current, lastDigit, tend;
long long fact, sum;
printf ("Enter upper limit: ");
scanf ("%d", &tend);
printf ("All Strong numbers between 1 to %d are: \n", tend);
for (k=1; k<=tend; k++)
{
current = k;
sum = 0;
while (current > 0)
{
fact = 1ll;
lastDigit = current % 10;
for (l=1; l<=lastDigit; l++)
{
fact = fact * l;
}
sum += fact;
current /= 10;
}
if(sum == k)
{
printf ("%d, ", k);
}
}
return 0;
}
```

Output:

Conclusion

From all the above examples and scenarios, it can be easily concluded that strong numbers must have both computations of sum and factorial mandatorily then only it will result in the original or actual number i.e. a strong number.

Recommended Articles

This is a guide to Strong Number in C. Here we discuss how to check Strong Number in C using various methods and examples along with its code implementation. You can also go through our other suggested articles to learn more –

1. Prime Numbers in C (https://www.educba.com/prime-numbers-in-c/)
2. Reverse Number in C (https://www.educba.com/reverse-number-in-c/)
3. Reverse String in C (https://www.educba.com/reverse-string-in-c/)
4. Prime Numbers in Java (https://www.educba.com/prime-numbers-in-java/)

(https://www.educba.com/software-development/)

About Us

Blog (https://www.educba.com/blog/?source=footer)

Who is EDUCBA? (https://www.educba.com/about-us/?source=footer)

Sign Up (https://www.educba.com/software-development/signup/?source=footer)

Corporate Training (https://www.educba.com/corporate/?source=footer)

Certificate from Top Institutions (https://www.educba.com/educbalive/?source=footer)

Contact Us (https://www.educba.com/contact-us/?source=footer)

Verifiable Certificate (https://www.educba.com/software-development/verifiable-certificate/?source=footer)

Reviews (https://www.educba.com/software-development/reviews/?source=footer)

Terms and Conditions (https://www.educba.com/terms-and-conditions/?source=footer)

Privacy Policy (https://www.educba.com/privacy-policy/?source=footer)

Apps

iPhone & iPad (https://itunes.apple.com/in/app/educba-learning-app/id1341654580?mt=8)

Android (https://play.google.com/store/apps/details?id=com.educba.www)

QUIZ

Resources

Free Courses (https://www.educba.com/software-development/free-courses/?source=footer)

Java Tutorials (https://www.educba.com/software-development/software-development-tutorials/java-tutorial/?source=footer)

Python Tutorials (https://www.educba.com/software-development/software-development-tutorials/python-tutorial/?source=footer)

All Tutorials (https://www.educba.com/software-development/software-development-tutorials/?source=footer)

Certification Courses

All Courses (https://www.educba.com/software-development/courses/?source=footer)

Software Development Course - All in One Bundle (https://www.educba.com/software-development/courses/software-development-course/?source=footer)

Become a Python Developer (https://www.educba.com/software-development/courses/python-certification-course/?source=footer)

Java Course (https://www.educba.com/software-development/courses/java-course/?source=footer)

Become a Selenium Automation Tester (https://www.educba.com/software-development/courses/selenium-training-certification/?source=footer)

Become an IoT Developer (https://www.educba.com/software-development/courses/iot-course/?source=footer)

ASP.NET Course (https://www.educba.com/software-development/courses/asp-net-course/?source=footer)

VB.NET Course (https://www.educba.com/software-development/courses/vb-net-course/?source=footer)

PHP Course (https://www.educba.com/software-development/courses/php-course/?source=footer)

QUIZ