



# CRACKING CODING INTERVIEW



## Recursion

প্রোগ্রামিং ভাষায় Recursion খুবই জনপ্রিয় একটি ফাংশন। যে ফাংশনটি নিজেকে কল করতে পারে। Recursion প্রোগ্রামিং ভাষায় কীভাবে প্রয়োগ করা যায়, এই ফাংশনটির সুবিধা-অসুবিধা নিয়েই আমরা আজকে আলোচনা করবো।

একটি recursive function- এর একটি base case এবং একটি recursive condition থাকে, এবং এই কোডটি বারবার কল হতেই থাকে। এখানে, base case টি কোডটি তার শর্ত বা condition পূরণ হয়েছে কিনা সেটি বুঝতে সাহায্য করে এবং recursive condition টি কোডটিকে বারবার call করার কাজে ব্যবহৃত হয়ে থাকে। Recursive function টি আজীবন চলতেই থাকবে, যদি base case এর condition টি পূরণ না হয়।

চলো আমরা C++ এর কোডে Recursion-এর প্রয়োগ দেখে আসি,

```
void recursion_function()
{
    recursion_function(); //Calling self
}
```

অথবা, নিচের কোডটিও ব্যবহার করা যেতে পারে,

```
void recurse() {
    recurse(); //Calling self
}
```

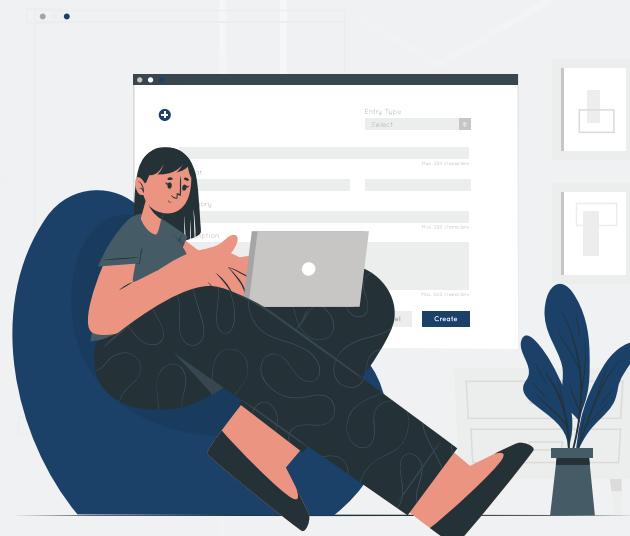
```
int main() {
    recurse();
}
```

নিচের চিত্রটি তোমাকে এই ফাংশনটি কীভাবে লুপের মধ্যে চলতেই থাকে তা বু�তে সাহায্য করবে,

```
void recurse() {
    recurse();           ← Recursive call
}

int main() {
    recurse();          ← Function call
}
```

এই Recursion ফাংশনটি তার রিকার্সিভ শর্তটি পূরণ করার পূর্ব পর্যন্ত চলতেই থাকবে। আর এই কাজটি একটি লুপ আকারে চলতেই থাকবে যতক্ষণ না পর্যন্ত চূড়ান্ত সমাধানটি খুঁজে না পাওয়া যায়। আর যখনই চূড়ান্ত সমাধানটির কোনো অংশ খুঁজে পাওয়া যাবে, তখন ফাংশনটিকে সেটিকে স্ট্যাক ডেটা স্ট্রাকচার ফরমেটে স্টোর করে রাখবে। যেগুলো চূড়ান্ত সমাধানটি খুঁজে বের করতে ব্যবহার করা হয়।



## সি ++ এ Recursion এর প্রকারভেদ

সি++ ভাষায় দুই ধরনের Recursion ফাংশন রয়েছে। তা হলো -

### (১) Direct Recursion

এখানে ফাংশনটি নিজেকে সরাসরি কল করে।

```
void direct_recursion()
{
    ...
    direct_recursion()
    ...
}
```

### (২) Indirect Recursion

এখানে ফাংশনটি নিজেকে সরাসরি কল না করে, অন্য একটি ফাংশনের মাধ্যমে কল করে থাকে।

```
void indirect_recursion() {
    ...
    func();
    ...
}
void func() {
    ...
    indirect_recursion();
    ...
}
```

## সি++ এ Recursion-এর উদাহরণ



চলো আমরা সি++ এ কীভাবে Recursion প্রয়োগ করতে পারি, তার কয়েকটি উদাহরণ দেখে নেই।

### Fibonacci Series Using Recursion

Fibonacci Series এমন একটি ক্রমিক সিরিজ, যেখানে প্রতিটি সংখ্যা তার পূর্বের দুটি সংখ্যার যোগফলের সমান। প্রথম দুটি সংখ্যাকে যদি আমরা ০ এবং ১ ধরি, তাহলে সিরিজটি হবে নিম্নরূপ-

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

চলো এখন আমরা দেখে নেই, আমরা কীভাবে এই সিরিজটি Recursion ব্যবহার করে তৈরি করতে পারি,

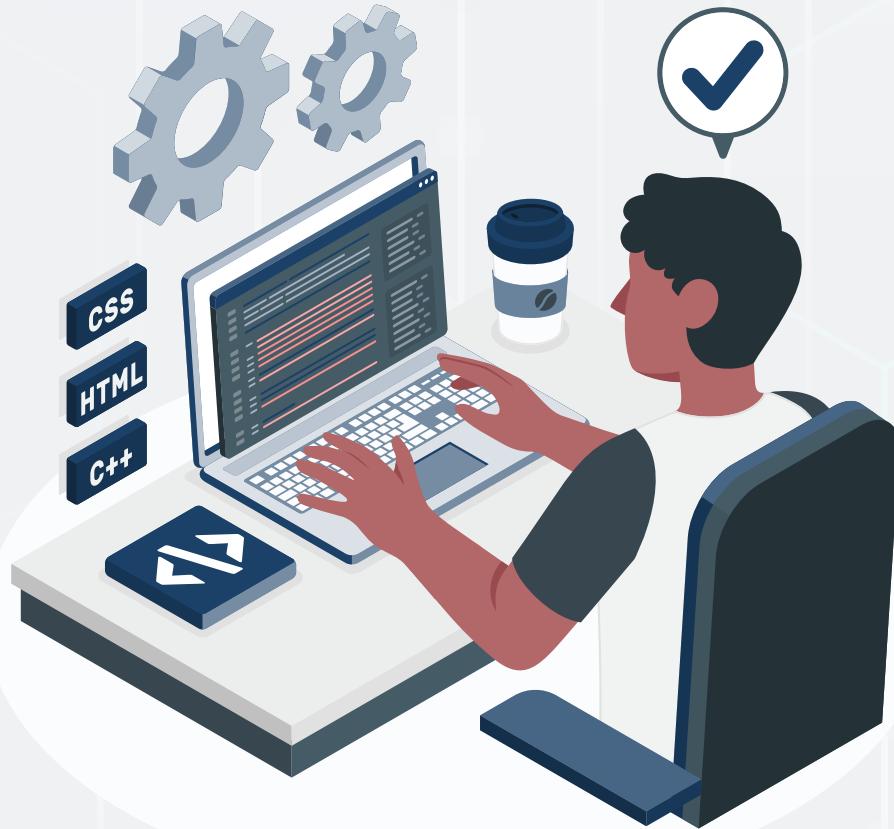
```
#include <iostream>
using namespace std;

int fib(int x) {
    if((x==1)|| (x==0)) {
        return(x);
    }
    else {
        return(fib(x-1) + fib(x-2));
    }
}
int main() {
    int x , i = 0;
    cout << "Enter the number of terms of
series : ";
    cin >> x;
    cout << "\nFibonacci Series : ";
    while(i < x) {
        cout << " " << fib(i);
        i++;
    }
    return 0;
}
```

আউটপুট

```
Enter the number of terms of series : 9
Fibonacci series : 0 1 1 2 3 5 8 13 21
```

উপরের প্রোগ্রামটিতে, রিকার্সিভ ফাংশনটি হলো, fib(), যেটি নিজেকে কল করতে থাকে এবং পুরো সিরিজটি তৈরি করে।



## Factorial Program Using Recursion

কোনো নাস্বারের Factorial বলতে সেই সংখ্যার সাথে তার থেকে ছোট এবং শূন্য থেকে বড় সকল মূর্ণ সংখ্যার গুনফলকে বোঝায়। তাহলে, আমরা যদি Factorial of 6 বলি। তাহলে তার উত্তর হবে নিম্নরূপ -

$$6! = 6 * 5 * 4 * 3 * 2 * 1 = 120$$

চলো এখন আমরা দেখে নেই, আমরা কীভাবে এই কাজটি Recursion ব্যবহার করে তৈরি করতে পারি,

```
#include <iostream>
using namespace std;

int factorial(int n);
int main()
{
    int n;
    cout << "Enter a positive integer: ";
    cin >> n;
    cout << "Factorial of " << n << " = " << factorial(n);
    return 0;
}

int factorial(int n)
{
    if(n > 1)
        return n * factorial(n - 1);
    else
        return 1;
}
```

## আউটপুট

```
Enter a positive integer: 8
Factorial of 8 = 40320
```

উপরের প্রোগ্রামটিতে, Recursive function-টি হলো `factorial()`, যা একটি মূর্ণসংখ্যা ইনপুট নেয় এবং রিকার্সিভ ফাংশনটির শর্তপূরণ হওয়ার আগ পর্যন্ত চলতেই থাকে। (i.e,  $n = 1$ )

পরিশেষে বলা যায় যে, প্রোগ্রামিং ভাষায় রিকার্সিভ ফাংশনটি খুবই কার্যকারী এবং জরুরি একটি ফাংশন। তাই তুমি যদি প্রোগ্রামিং ভাষার দক্ষতা বৃদ্ধি করতে চান তাহলে অবশ্যই এই ফাংশনটির দক্ষতা অর্জন করা উচিত।

