# C++ Recursion

A function that calls itself is known as a recursive function. And, this technique is known as recursion.

Any recursive definition has two parts:

1. **Base**
2. **Recursion**

**Base:** An initial simple definition which can not be expressed in terms of smaller version.
**Recursion:** The part of the definition which can be expressed in terms of smaller versions of itself.

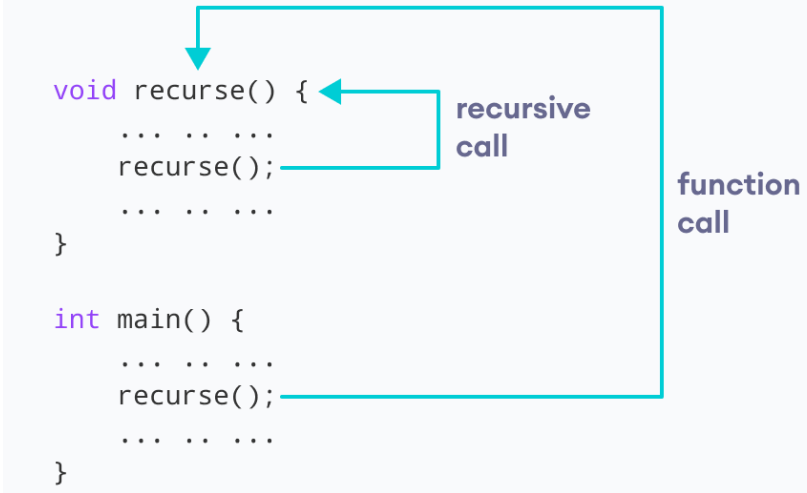
---

## Working of Recursion in C++

```
void recurse()
{
    ... .. ...
    recurse();
    ... .. ...
}

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

The figure below shows how recursion works by calling itself over and over again.




How recursion works in C++ programming

The recursion continues until some condition is met.

To prevent infinite recursion, if...else statement (or similar approach) can be used where one branch makes the recursive call and the other doesn't.

---

## Example 1: Factorial of a Number Using Recursion

```
// Factorial of n = 1*2*3*...*n

#include <iostream>
using namespace std;

int factorial(int);

int main() {
    int n, result;

    cout << "Enter a non-negative number: ";
    cin >> n;

    result = factorial(n);
    cout << "Factorial of " << n << " = " << result;
    return 0;
}
```

```
int factorial(int n) {
    if (n <= 1) {
        return 1;
     else
        return n * factorial(n - 1);

}
```

## Output

```
Enter a non-negative number: 4
Factorial of 4 = 24
```

---

**Two Musts for Recursion**

- Base case is a must, that makes no recursive calls
- When you make a recursive call it should be to a simpler instance and make forward progress towards the base case.

**Summary:**

- Break a problem into smaller subproblems of the same form and call the same function again on that smaller problem.
- Powerful programming tool
- Not always a wise choice, but often a good one
- Some problems are solved easily by recursion than by iteration