

Recursion

Dr. Asif Uddin Khan

C Recursion

- A function that calls itself is known as a recursive function. And, this technique is known as recursion.
- The recursion continues until some condition is met to prevent it.

How does recursion work?

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

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

The diagram shows two function definitions. The first is `void recurse()` with a body containing three lines: `... ..`, `recurse();`, and `... ..`. The second is `int main()` with a body containing three lines: `... ..`, `recurse();`, and `... ..`. A line connects the `recurse();` line in `main()` to the `recurse()` line in the `recurse()` function. Another line connects the `recurse();` line inside the `recurse()` function back to the `recurse()` line in the `recurse()` function. A bracket on the right side of these two lines is labeled "recursive call".

C Recursion

- To prevent infinite recursion, [if...else statement](#) (or similar approach) can be used where one branch makes the recursive call, and other doesn't.
- A recursive function calls itself with smaller size argument.
- There should be some terminating condition
- The recursion continues until the condition is met.
- To prevent infinite recursion, [if...else statement](#) (or similar approach) can be used where one branch makes the recursive call, and other doesn't.

factorial using recursion

```
//factorial using recursion
#include <stdio.h>
int fact (int);
int main()
{
    int n,f;
    printf("Enter the number");
    scanf("%d",&n);
    f = fact(n);
    printf("factorial = %d",f);
}
```

```
int fact(int n)
{
    if (n==0)
    {
        return 1;
    }
    else if ( n == 1)
    {
        return 1;
    }
    else
    {
        return n*fact(n-1);
    }
}
```

factorial using recursion

return 5 * factorial(4) = 120

└ return 4 * factorial(3) = 24

└ return 3 * factorial(2) = 6

└ return 2 * factorial(1) = 2

└ return 1 * factorial(0) = 1

javaTpoint.com

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

Fig: Recursion

Example: Sum of Natural Numbers Using Recursion

```
//sum of natural numbers using
//recursion
#include <stdio.h>
int sum(int n);
int main() {
    int number, result;
    printf("Enter a positive integer: ");
    scanf("%d", &number);
    result = sum(number);
    printf("sum = %d", result);
    return 0;
}
int sum(int n) {
    if (n != 0)
        // sum() function calls itself
        return n + sum(n-1);
    else
        return n;
}
```

