

# Recursion

(1) - End ?

→ Condition (Stop) ?

\* Base Condition

→ Solution Available  
(known) -

void func()

{

↓ ↓ ↓ ↓

Cout << "A function" << endl;  
func();

↑↑↑

main

func();

↙

int factorial(int num)

{ if (num == 0 || num == 1) // Base Case

return 1;

else

return num \* factorial(num - 1);

}

factorial(4) num = 4 ↗ 24

return 4 \* factorial(3)

↗ num = 3

return 3 \* factorial(2)

factorial(4)

num = 2 ↗ 2

return 2 \* factorial(1)

factorial(3)

Table of n? Solution?  $n \times 1 = \sqrt{n}$

void tableOf(int val, int start, int till)

{ if (start > till) // Base Condition  
return;

$2 \times 1 = 2$

$2 \times 2 = 4$

$2 \times 3 = 6$

$2 \times 4 = 8$

$2 \times 5 = 10$

else

1,5) {

Cout << val << " \* " << start << "=" << val \* start;

Cout << endl;

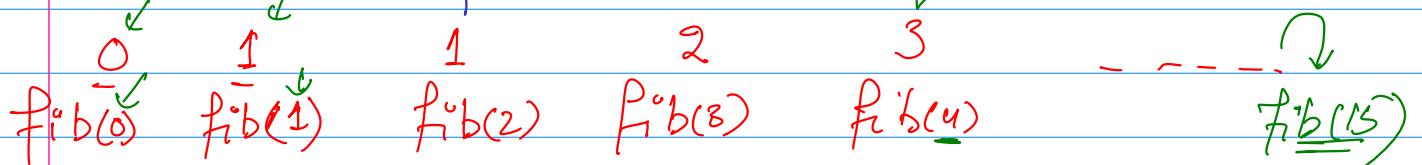
tableOf(val, start + 1, till);

}

tableOf(2)

## Fibonacci Sequences

Golden Spiral & Fibonacci Sequence.



$$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$$

```
int fib(int n)
```

```
if (n == 0 || n == 1) // Base Case
    return n;
```

else

```
    return fib(n-1) + fib(n-2);
```

```
fib(4) → return fib(3) + fib(2) → return fib(1) + fib(0)
```

$\begin{matrix} 3 \\ 2 \\ 1 \end{matrix} + \begin{matrix} 1 \\ 1 \end{matrix} = 3$

$\begin{matrix} 1 \\ 1 \end{matrix} + \begin{matrix} 1 \\ 1 \end{matrix} = 2$

$\begin{matrix} 1 \\ 1 \end{matrix} + \begin{matrix} 0 \\ 0 \end{matrix} = 1$

Slow Recursive methods