Recursividad

M.S.C. Jacob Green

Recursividad

- Es una forma de especificar un proceso en base a su propia definición.

- Es una estructura alternativa realizar tareas repetitivas (ciclos).

- Se llama función recursiva a aquella que se llama así misma y consta de dos partes:
  - Caso base: ocasiona que la función deje de llamarse así misma.
  - Caso recursivo: implica volver a llamar a la función, utilizando parámetros cada vez más cercanos al caso base.

Factorial

- El factorial de un número implica multiplicar todos los números desde 1 hasta n.

- Ejemplo:
  - 5! = 5 * 4 * 3 * 2 * 1 = 120

Factorial

- El factorial de un número implica multiplicar todos los números desde 1 hasta n.

- Ejemplo:
  - 5! = 5 * 4 * 3 * 2 * 1 = 120

- La definición matemática de factorial es:
  - n! = n * (n - 1)!
  - Es decir, n! = n * (n-1) * (n-2) * (n-3) … * 1

Factorial

- El factorial de un número implica multiplicar todos los números desde 1 hasta n.

- Ejemplo:
  - 5! = 5 * 4 * 3 * 2 * 1 = 120

- La definición matemática de factorial es:
  - n! = n * (n - 1)!
  - Es decir, n! = n * (n-1) * (n-2) * (n-3) … * 1

```
int factorial(int n) {
    int fact = 1;
    while(n > 1) {
        fact = fact * n;
        n = n - 1;
    }
    return fact;
}
```

Factorial

```
int factorial(int n) {
    int fact = 1;
    while(n > 1) {
        fact = fact * n;
        n = n - 1;
    }
    return fact;
}
```

Iteración	n	fact	n actualizada
1	5	1 * 5 = 5	5 - 1 = 4

Factorial

```
int factorial(int n) {
    int fact = 1;
    while(n > 1) {
        fact = fact * n;
        n = n - 1;
    }
    return fact;
}
```

Iteración	n	fact	n actualizada
1	5	1 * 5 = **5**	5 - 1 = **4**
2	**4**	**5** * 4 = 20	4 - 1 = 3

Factorial

```
int factorial(int n) {
    int fact = 1;
    while(n > 1) {
        fact = fact * n;
        n = n - 1;
    }
    return fact;
}
```

Iteración	n	fact	n actualizada
1	5	1 * 5 = 5	5 - 1 = 4
2	4	5 * 4 = **20**	4 - 1 = **3**
3	**3**	**20** * 3 = 60	3 - 1 = 2

Factorial

```
int factorial(int n) {
    int fact = 1;
    while(n > 1) {
        fact = fact * n;
        n = n - 1;
    }
    return fact;
}
```

Iteración	n	fact	n actualizada
1	5	1 * 5 = 5	5 - 1 = 4
2	4	5 * 4 = 20	4 - 1 = 3
3	3	20 * 3 = **60**	3 - 1 = **2**
4	**2**	**60** * 2 = 120	2 - 1 = 1

Factorial

```
int factorial(int n) {
    int fact = 1;
    while(n > 1) {
        fact = fact * n;
        n = n - 1;
    }
    return fact;
}
```

Iteración	n	fact	n actualizada
1	5	1 * 5 = 5	5 - 1 = 4
2	4	5 * 4 = 20	4 - 1 = 3
3	3	20 * 3 = 60	3 - 1 = 2
4	2	60 * 2 = 120	2 - 1 = **1**
5	**1**	------	------

Factorial

```
int factorial(int n) {
    int fact = 1;
    while(n > 1) {
        fact = fact * n;
        n = n - 1;
    }
    return fact;
}
```

Iteración	n	fact	n actualizada
1	5	1 * 5 = 5	5 - 1 = 4
2	4	5 * 4 = 20	4 - 1 = 3
3	3	20 * 3 = 60	3 - 1 = 2
4	2	60 * 2 = **120**	2 - 1 = 1
5	1	------	------

el factorial de **n = 5** es **120**

Factorial

- También se puede definir de la siguiente manera:

$$n! = \begin{cases} 1 & \text{si, } n = 0 \\ (n-1)! * n & \text{si, } n >= 1 \end{cases}$$

Factorial

- También se puede definir de la siguiente manera:

$$n! = \begin{cases} 1 & \text{si, } n = 0 \quad \leftarrow \textbf{caso base} \\ (n-1)! * n & \text{si, } n >= 1 \quad \leftarrow \textbf{caso recursivo} \end{cases}$$

Factorial

- También se puede definir de la siguiente manera:

$$n! = \begin{cases} 1 & \text{si, } n = 0 \quad \leftarrow \text{caso base} \\ (n-1)! * n & \text{si, } n >= 1 \quad \leftarrow \text{caso recursivo} \end{cases}$$

Definición recursiva!!!

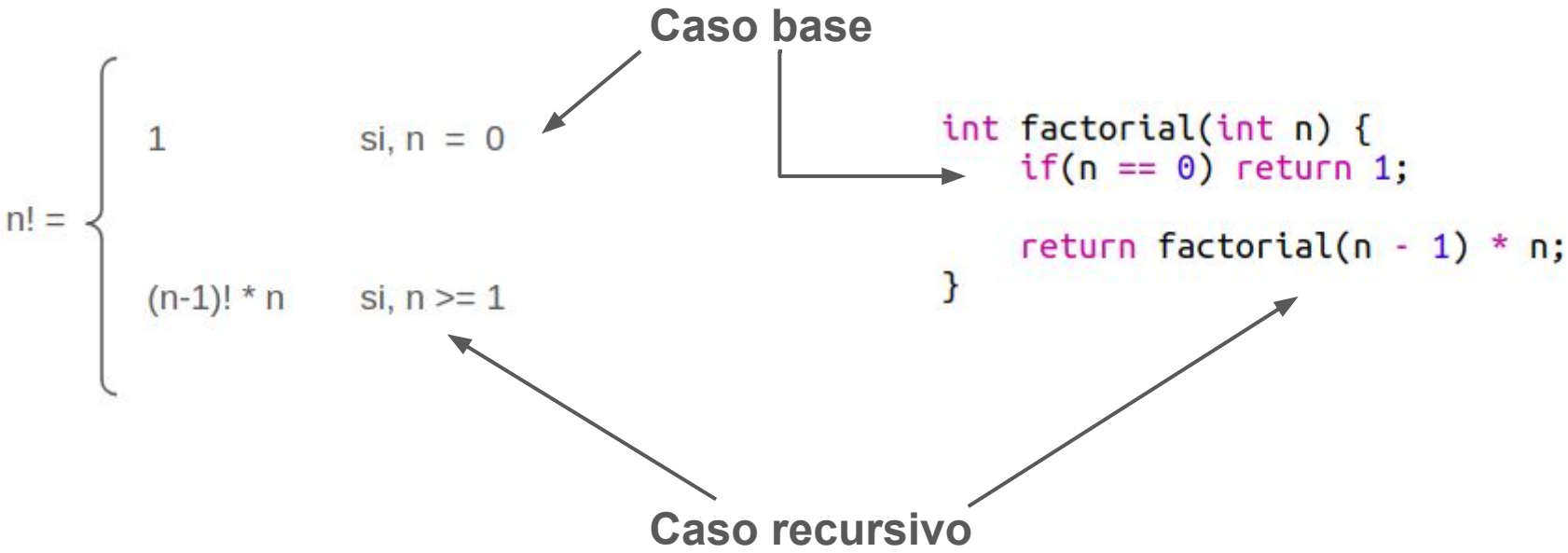
Factorial

$$n! = \begin{cases} 1 & \text{si, } n = 0 \\ (n-1)! * n & \text{si, } n >= 1 \end{cases}$$

```c
int factorial(int n) {
    if(n == 0) return 1;

    return factorial(n - 1) * n;
}
```

Factorial


Caso base
n! =
1 si, n = 0
(n-1)! * n si, n >= 1
int factorial(int n) {
if(n == 0) return 1;
return factorial(n - 1) * n;
}
Caso recursivo

Factorial

```
int factorial(int n) {
    if(n == 0) return 1;

    return factorial(n - 1) * n;
}
```

Llamada	n	valor parcial	return
1	**5**	?	factorial(4) * **5**

Factorial

```
int factorial(int n) {
    if(n == 0) return 1;

    return factorial(n - 1) * n;
}
```

Llamada	n	valor parcial	return
1	5	?	factorial(4) * 5
2	**4**	?	factorial(**3**) * **4**

Factorial

```
int factorial(int n) {
    if(n == 0) return 1;

    return factorial(n - 1) * n;
}
```

Llamada	n	valor parcial	return
1	5	?	factorial(4) * 5
2	4	?	factorial(3) * 4
3	**3**	?	factorial(**2**) * **3**

Factorial

```
int factorial(int n) {
    if(n == 0) return 1;

    return factorial(n - 1) * n;
}
```

Llamada	n	valor parcial	return
1	5	?	factorial(4) * 5
2	4	?	factorial(3) * 4
3	3	?	factorial(2) * 3
4	**2**	?	factorial(**1**) * **2**

Factorial

```
int factorial(int n) {
    if(n == 0) return 1;

    return factorial(n - 1) * n;
}
```

Llamada	n	valor parcial	return
1	5	?	factorial(4) * 5
2	4	?	factorial(3) * 4
3	3	?	factorial(2) * 3
4	2	?	factorial(1) * 2
5	1	?	factorial(0) * 1

Factorial

```
int factorial(int n) {
    if(n == 0) return 1;

    return factorial(n - 1) * n;
}
```

Llamada	n	valor parcial	return
1	5	?	factorial(4) * 5
2	4	?	factorial(3) * 4
3	3	?	factorial(2) * 3
4	2	?	factorial(1) * 2
5	1	?	factorial(0) * 1
6	**0**	**1**	**1**

Factorial

```
int factorial(int n) {
    if(n == 0) return 1;

    return factorial(n - 1) * n;
}
```

Llamada	n	valor parcial	return
1	5	?	factorial(4) * 5
2	4	?	factorial(3) * 4
3	3	?	factorial(2) * 3
4	2	?	factorial(1) * 2
5	1	**1** * 1 = 1	**factorial(0)** * 1
6	0	**1**	1

Factorial

```
int factorial(int n) {
    if(n == 0) return 1;

    return factorial(n - 1) * n;
}
```

Llamada	n	valor parcial	return
1	5	?	factorial(4) * 5
2	4	?	factorial(3) * 4
3	3	?	factorial(2) * 3
4	2	**1** * 2 = 2	**factorial(1)** * 2
5	1	1 * 1 = **1**	factorial(0) * 1
6	0	1	1

Factorial

```
int factorial(int n) {
    if(n == 0) return 1;

    return factorial(n - 1) * n;
}
```

Llamada	n	valor parcial	return
1	5	?	factorial(4) * 5
2	4	?	factorial(3) * 4
3	3	**2** * 3 = 6	**factorial(2)** * 3
4	2	1 * 2 = **2**	factorial(1) * 2
5	1	1 * 1 = 1	factorial(0) * 1
6	0	1	1

Factorial

```
int factorial(int n) {
    if(n == 0) return 1;

    return factorial(n - 1) * n;
}
```

Llamada	n	valor parcial	return
1	5	?	factorial(4) * 5
2	4	**6** * 4 = 24	**factorial(3)** * 4
3	3	2 * 3 = **6**	factorial(2) * 3
4	2	1 * 2 = 2	factorial(1) * 2
5	1	1 * 1 = 1	factorial(0) * 1
6	0	1	1

Factorial

```
int factorial(int n) {
    if(n == 0) return 1;

    return factorial(n - 1) * n;
}
```

Llamada	n	valor parcial	return
1	5	**24** * 5 = 120	**factorial(4)** * 5
2	4	6 * 4 = **24**	factorial(3) * 4
3	3	2 * 3 = 6	factorial(2) * 3
4	2	1 * 2 = 2	factorial(1) * 2
5	1	1 * 1 = 1	factorial(0) * 1
6	0	1	1

Factorial

```
int factorial(int n) {
    if(n == 0) return 1;

    return factorial(n - 1) * n;
}
```

El factorial de **n = 5** es **120**

Llamada	n	valor parcial	return
1	5	24 * 5 = **120**	factorial(4) * 5
2	4	6 * 4 = 24	factorial(3) * 4
3	3	2 * 3 = 6	factorial(2) * 3
4	2	1 * 2 = 2	factorial(1) * 2
5	1	1 * 1 = 1	factorial(0) * 1
6	0	1	1