FUNDAMENTOS DE PROGRAMACIÓN - CÁTEDRA MENDEZ

RECURSIVIDAD

¿QUÉ ES?

¿PARA QUÉ SIRVE?



¿QUÉ SE IMAGINAN CUÁNDO ESCUCHAN "RECURSIVIDAD"?

No hay respuestas incorrectas

ALGO ES RECURSIVO SI SU DEFINICIÓN LO CONTIENE A SÍ MISMO.

RESOLVER UN PROBLEMA USANDO LA SOLUCIÓN DE UNA VERSIÓN MÁS CHICA DEL MISMO PROBLEMA.

MISMO PROBLEMA



PROBLEMA MÁS CHICO

PRIMER PROBLEMA

HOMERO TIENE 5 DONAS

¿PODRÍAMOS EXPLICAR CÓMO SE LAS COME PENSANDO SOLO EN UNA DONA A LA VEZ?



PENSEMOS EL PROBLEMA

HOMERO TIENE QUE COMER 5 DONAS

↓

HOMERO COME 1

↓

HOMERO TIENE QUE COMER 4 DONAS

PENSEMOS EL PROBLEMA

HOMERO COME 1

↓

HOMERO TIENE QUE COMER 3 DONAS

↓

HOMERO COME 1

PENSEMOS EL PROBLEMA

ESTAMOS RESOLVIENDO
SIEMPRE EL MISMO
PROBLEMA...
PERO CADA VEZ MÁS CHICO.

ANATOMÍA DE UNA FUNCIÓN RECURSIVA

TODA FUNCIÓN RECURSIVA TIENE 3 PARTES

1. CONDICIÓN DE CORTE
2. PROCESAMIENTO
3. LLAMADO RECURSIVO

NUESTRA PRIMERA FUNCIÓN RECURSIVA

```c
void comer_donas(int cantidad){

    if (cantidad == 0){
        return;
    }

    printf("Homero tiene %i donas y se come una\n", cantidad);

    comer_donas(cantidad - 1);
}
```

CASO BASE (CONDICIÓN DE CORTE)

EL CASO BASE ES EL CASO EN EL QUE EL PROBLEMA TIENE SOLUCIÓN DIRECTA Y NO PUEDE DIVIDIRSE EN MÁS SUBPROBLEMAS.

```
void comer_donas(int cantidad) {

    if(cantidad == 0){
        return;
    }
```

PROCESAMIENTO

¿QUÉ HAGO EN ESTA LLAMADA?

CADA LLAMADA HACE UNA PEQUEÑA PARTE DEL TRABAJO.

```
printf("Homero tiene %i donas y se come una\n", cantidad);
```



LLAMADO RECURSIVO

¿CÓMO HAGO EL PROBLEMA MÁS CHICO?

```
comer_donas(cantidad - 1);
```

5 → 4 → 3 → 2 → 1 → 0

LLAMADO RECURSIVO

PARA VOLVER A REALIZAR LA MISMA RESOLUCIÓN PERO CON UN PROBLEMA MÁS CHICO LA FUNCIÓN NECESITA LLAMARSE A SÍ MISMA CON ALGÚN PARÁMETRO DISTINTO.

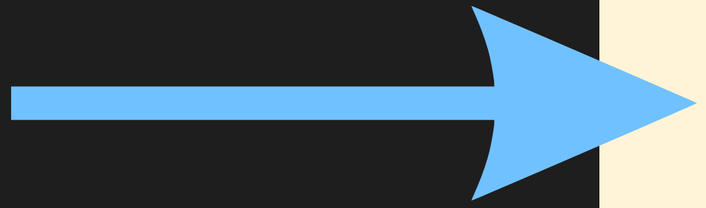
NUESTRA PRIMERA FUNCIÓN RECURSIVA

```c
void comer_donas(int cantidad) {

    if(cantidad == 0){
        return;
    }

    printf("Homero come una dona\n");

    comer_donas(cantidad - 1);
}
```



CASO BASE



PROCESAMIENTO



LLAMADO RECURSIVO

CASO BASE (CONDICIÓN DE CORTE)

¿QUÉ PASA SI NO TENGO UN CASO BASE?

PENSEMOS EL PROBLEMA

```c
void comer_donas(int cantidad) {

    printf("Homero come una dona\n");

    comer_donas(cantidad - 1);
}
```

PENSEMOS EL PROBLEMA

¿FUNCIONA?

BART, TENEMOS UN PROBLEMA



NUNCA TERMINA 😬

COMER_DONAS(5)
COMER_DONAS(4)
COMER_DONAS(3)
COMER_DONAS(2)
COMER_DONAS(1)
COMER_DONAS(0)
COMER_DONAS(-1)
COMER_DONAS(-2)
...

CASO BASE (CONDICIÓN DE CORTE)

EL CASO BASE ES LA CONDICIÓN DE CORTE DE LA RECURSIVIDAD, Y LO QUE EVITA QUE SE EJECUTE INFINITAMENTE.

LLAMADO RECURSIVO

¿QUÉ PASA SI NO ACHICAMOS EL PROBLEMA?

LLAMADO RECURSIVO

```c
void comer_donas(int cantidad){

    if (cantidad == 0){
        return;
    }

    printf("Homero come una dona\n");

    comer_donas(cantidad);
}
```

COMER_DONAS(5)



COMER_DONAS(5)



COMER_DONAS(5)

...

LLAMADO RECURSIVO

```
comer_donas(cantidad);        // ❌

comer_donas(cantidad - 1);    // ✅
```

EL LLAMADO RECURSIVO TIENE QUE ACERCARNOS A LA CONDICIÓN DE CORTE

PARA HACER RECURSIVIDAD, PREGUNTATE:

¿CUÁNDO TERMINO?

¿QUÉ HAGO YO?

¿CÓMO ACHICO EL PROBLEMA?

EJEMPLOS

$$5! = 5 \cdot 4!$$
$$4! = 4 \cdot 3!$$
$$3! = 3 \cdot 2!$$
$$2! = 2 \cdot 1!$$
$$1! = 1$$
$$0! = 1$$

FACTORIAL

MISMO PROBLEMA



PROBLEMA MÁS CHICO

EJEMPLOS

FACTORIAL

OJO: LAS TRES PARTES NO SIEMPRE APARECEN EN TRES LÍNEAS DISTINTAS

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

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

EJEMPLOS

BÚSQUEDA LINEAL RECURSIVA

[4 | 7 | 2 | 9 | 5]

BUSQUEMOS EL NÚMERO 9

SIN USAR FOR NI WHILE, ¿CÓMO PUEDO BUSCAR UN ELEMENTO?

EJEMPLOS

PENSEMOS RECURSIVAMENTE CON LAS 3 PREGUNTAS

¿CUÁNDO TERMINO?

DOS CASOS BASE:

- ENCONTRÉ EL ELEMENTO
- LLEGUÉ AL FINAL DEL VECTOR

EJEMPLOS

¿QUÉ HAGO YO?

PROCESAMIENTO



COMPARO EL ELEMENTO
ACTUAL
CON EL ELEMENTO BUSCADO

EJEMPLOS

¿CÓMO ACHICO EL PROBLEMA?

LLAMADO RECURSIVO



AVANZO A LA SIGUIENTE POSICIÓN

EJEMPLOS

CÓDIGO

```c
#define MAXIMO 100

int buscar(int vector[MAXIMO], int tope, int buscado, int posicion_actual){

    if (posicion_actual == tope){
        return -1;
    }

    if (vector[posicion_actual] == buscado){
        return posicion_actual;
    }

    return buscar(vector, tope, buscado, posicion_actual + 1);

}
```

EJEMPLOS

¿QUÉ IMPRIME SI LLAMO MOSTRAR(3)?

```c
void mostrar(int n){

    if (n == 0){
        return;
    }

    printf("%i\n", n);

    mostrar(n-1);
}
```

3 2 1

1 2 3

Nada

EJEMPLOS

¿Y SI LLAMO A LA FUNCIÓN ANTES DE PROCESAR?

```c
void mostrar(int n){

    if (n == 0){
        return;
    }

    mostrar(n-1);

    printf("%i\n", n);
}
```

¿QUÉ SE IMAGINAN AHORA CUANDO ESCUCHAN "RECURSIVIDAD"?

¿CAMBIÓ ALGO RESPECTO DEL PRINCIPIO?