

Recursividad

Profesor Francisco Alejandro Medina

Objetivos

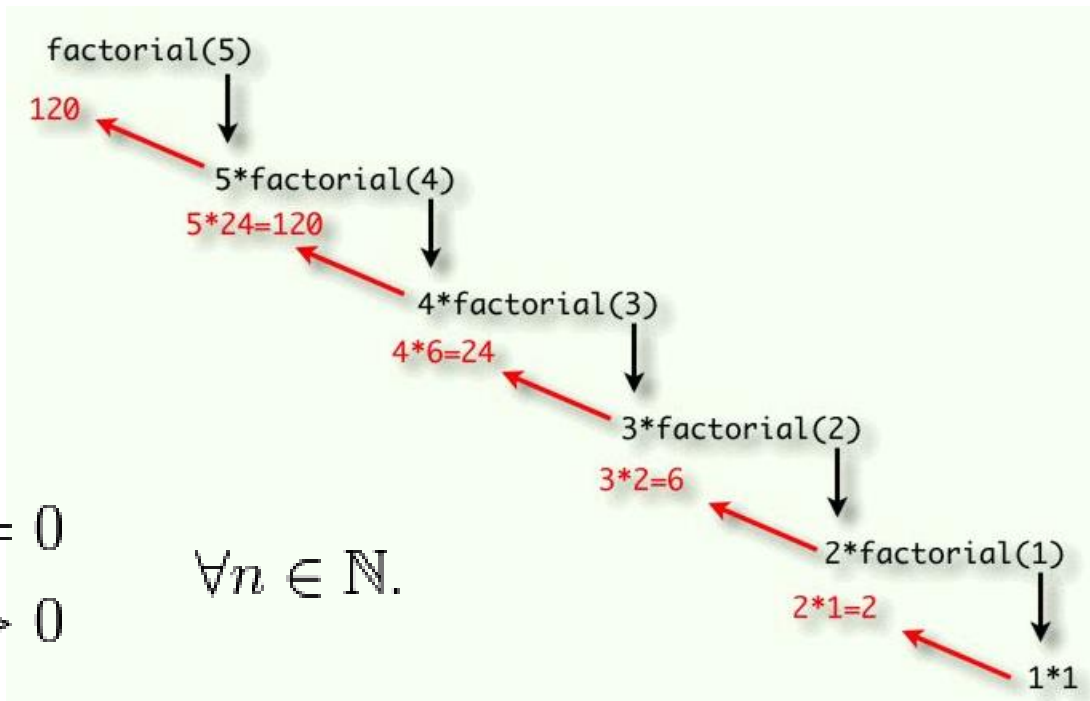
- Conocer los fundamentos del diseño de algoritmos recursivos.
- Comprender la ejecución de algoritmos recursivos.
- Aprender a realizar trazas de algoritmos recursivos.
- Comprender las ventajas e inconvenientes de la recursividad.

Que es Recursividad

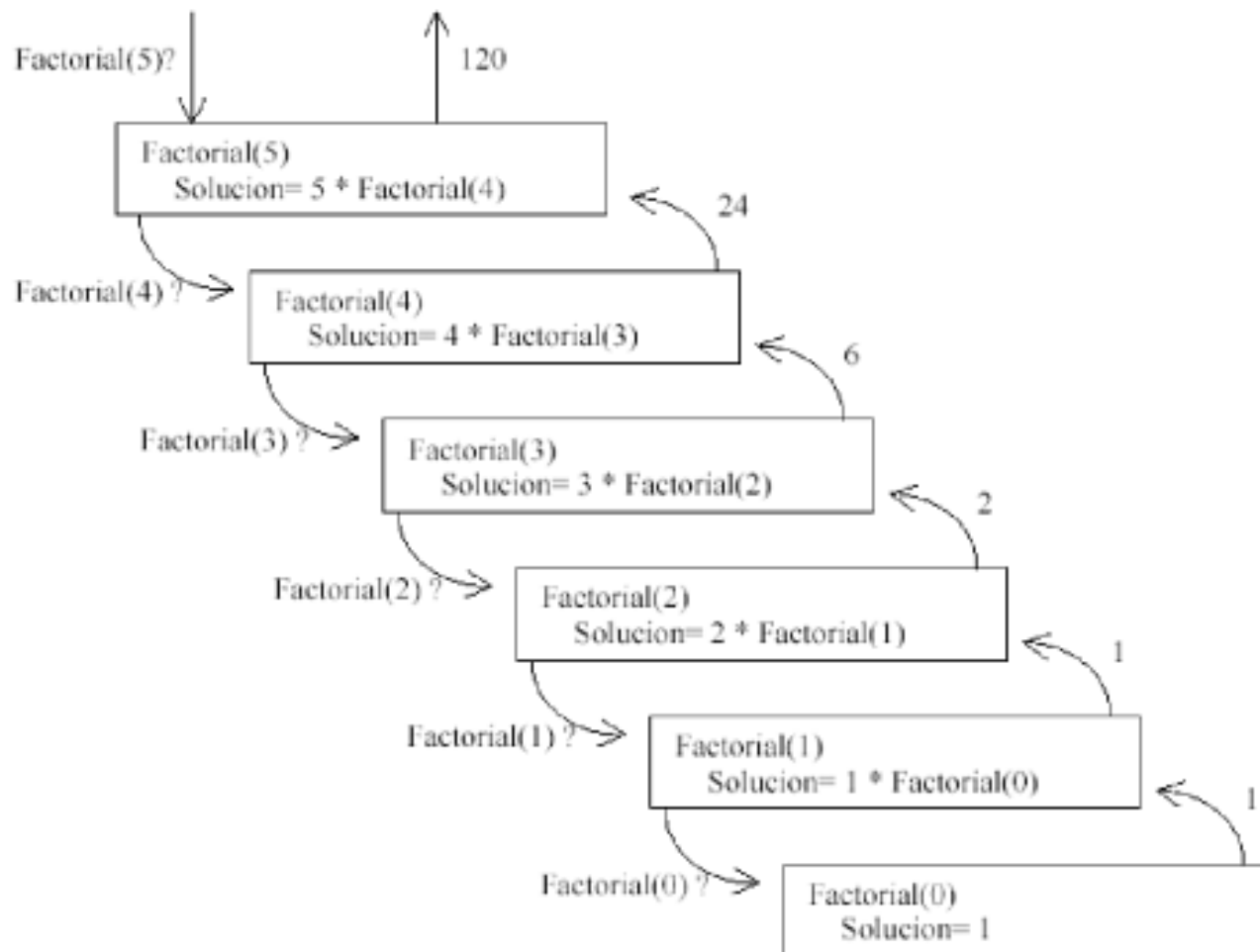
La recursividad es una técnica de programación importante. Se utiliza para realizar una llamada a una función desde la misma función.

Como ejemplo útil se puede presentar el cálculo de números factoriales.

$$n! = \begin{cases} 1 & \text{Si } n = 0 \\ n(n-1)! & \text{Si } n > 0 \end{cases} \quad \forall n \in \mathbb{N}.$$



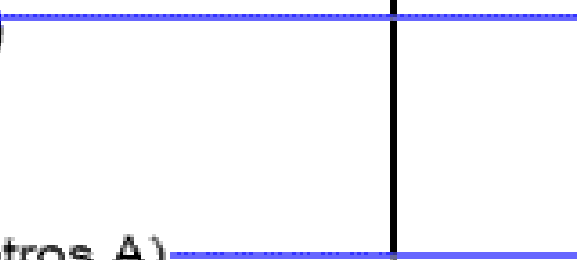
Ejemplo de Recursividad



Funciones Recursivas

Una función que se llama a sí misma se denomina **recursiva**

```
Metodo_A (parametros A)
{
    Sentencia 1
    Sentencia 2
    Metodo_A (parámetros A)
    .....
    .....
    Sentencia N
}
```



Ventajas y desventajas de la recursividad

La principal ventaja de la recursividad frente a los algoritmos iterativos es que se da lugar a algoritmos simples y compactos. La principal desventaja es que la recursividad resulta más lenta y consume más recursos al ejecutarse.

La razón por la que las funciones recursivas consumen más recursos que las iterativas es por la forma de resolverse las llamadas recursivas. Al ejecutarse una llamada de una función recursiva se almacena en la pila:

- Los argumentos de la función o subrutina.
- Las variables locales del subprograma
- La dirección de retorno, es decir el punto del programa que debe ejecutarse una vez acabe la llamada actual.

Ventajas y desventajas de la recursividad

- Las funciones recursivas son mecanismos muy eficientes de programación pues facilitan el diseño de las mismas, sin embargo la recursividad no es en todos los casos un buen modo de resolver problemas, ya que existen casos en los que un algoritmo iterativo resolvería de manera bastante adecuada un problema determinado.
- La recursividad consume recursos adicionales de memoria y tiempo de ejecución, y se debe aplicar a funciones que realmente obtengan beneficio directo.

Recursividad en Racket

Ejemplo: Se pide construir un programa en DrScheme tal que imprima las tablas de multiplicar del número 1 al 100, donde para cada una de ellas se multiplique del 1 al 10; así:

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10

2 * 1 = 2
2 * 2 = 2
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20

3 * .....
...
...

99 * 1 = 99
99 * 2 = 198
99 * 3 = 297
99 * 4 = 396
99 * 5 = 495
99 * 6 = 594
99 * 7 = 693
99 * 8 = 792
99 * 9 = 891
99 * 10 = 990

100 * 1 = 100
100 * 2 = 200
100 * 3 = 300
100 * 4 = 400
100 * 5 = 500
100 * 6 = 600
100 * 7 = 700
100 * 8 = 800
100 * 9 = 900
100 * 10 = 1000
```

1. Análisis

Se aprecia que lo que se requiere imprimir por cada renglón es lo siguiente:

1	*	1	=	(* 1 1)
↑	↑	↑	↑	↑
NÚMERO	CADENA	NUMERO	CADENA	EXPRESIÓN

Usaremos **V** y **r** para cada número; así:

V	*	r	=	(* V r)
↑	↑	↑	↑	↑
NÚMERO	CADENA	NUMERO	CADENA	EXPRESIÓN

De lo pedido vemos que el primer número (**V**) solo se incrementa en uno(1) cuando el número (**r**) ya ha tomado los valores del 1 al 10 (1 2 3 4 5 6 7 8 9 10)

V	r
1	1 2 3 4 5 6 7 8 9 10
2	1 2 3 4 5 6 7 8 9 10
...
9	1 2 3 4 5 6 7 8 9 10
10	1 2 3 4 5 6 7 8 9 10

Recursividad en Racket

De los ejercicios vistos en clase, podemos construir una función para la primera tabla, la den uno(1)

```
(define (m v r)
  (if (<= r 10)
      (begin
        (display v)
        (display " * ")
        (display r)
        (display " = ")
        (display (* v r))
        (newline)
        (m v (+ r 1)))
      )
  )
(m 1 1)
```

Lo que imprimiría

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10
```

Porque solo cambiamos el
valor de **r** en la función

Solo nos quedaría faltando una función que haga cambiar a **V** desde el 1 al 100, que también la sabemos hacer, iniciemos por una que imprima los números del 1 al 100; así:

```
(define (z V)
  (if (<= v 100)
      (begin
        (display v)
        (z (+ v 1))
      )
    )
  (z 1))
```

Tomando lo anterior, de lo que esta encerrado en el cuadro, rescatamos que aquí **V** va cambiando su valor de 1 a 100. Lo que haremos es utilizar este punto para meter allí la primera función definida; así:

```
(define (z v)
  (if (<= v 100)
      (begin
        (m v 1)
        (z (+ v 1))
      )
    )
  (z 1))
```

Recursividad en Racket

Ejercicio: Que imprime en pantalla el siguiente programa:

```
(define (m v r)
  (if (<= r 10)
      (begin
        (display v) (display " * ") (display r) (display "=") (display (* v r))
        (newline)
        (m v (+ r 1)))
      )
  )
)
(define (z v)
  (if (<= v 100)
      (begin
        (m v 1)
        (z (+ v 1))
      )
  )
)
(z 100)
```

Ejercicio: Que imprime en pantalla el siguiente programa:

```
(define (m v r)
  (if (<= r 10)
      (begin
        (display v) (display " * ") (display r) (display "=") (display (* v r))
        (newline)
        (m v (+ r 1)))
      )
  )
)
(define (z v)
  (if (<= v 100)
      (begin
        (m v 1)
        (z (+ v 1))
      )
  )
)
(z 99)
```

Ejercicio: Que imprime en pantalla el siguiente programa:

```
(define (m v r)
  (if (<= r 10)
      (begin
        (display v) (display " * ") (display r) (display "=") (display (* v r))
        (newline)
        (m v (+ r 1)))
      )
  )
)
(define (z v)
  (if (<= v 100)
      (begin
        (m v 10)
        (z (+ v 1))
      )
  )
)
(z 100)
```

Ejercicio: Que imprime en pantalla el siguiente programa:

```
(define (m v r)
  (if (<= r 10)
      (begin
        (display v) (display " * ") (display r) (display "=") (display (* v r))
        (newline)
        (m v (+ r 1)))
      )
  )
)
(define (z v)
  (if (<= v 100)
      (begin
        (m v 5)
        (z (+ v 1))
      )
  )
)
(z 99)
```

Recursividad en Racket

Ejercicio: Que imprime en pantalla el siguiente programa:

```
(define (m v r)
  (if (<= v 100)
      (if (<= r 10)
          (begin
            (display v) (display " * ") (display r) (display " = ") (display (* v r))
            (newline)
            (m v (+ r 1)))
          )
      (m (+ v 1) 1))
  )
)
(m 1 1)
```

Ejercicio: Que imprime en pantalla el siguiente programa:

```
(define (m v r)
  (if (<= v 100)
      (if (<= r 10)
          (begin
            (display v) (display " * ") (display r) (display " = ") (display (* v r))
            (newline)
            (m v (+ r 1)))
          )
      (m (+ v 1) 1))
  )
)
(m 99 1)
```

Ejercicio: Que imprime en pantalla el siguiente programa:

```
(define (m v r)
  (if (<= v 100)
      (if (<= r 10)
          (begin
            (display v) (display " * ") (display r) (display " = ") (display (* v r))
            (newline)
            (m v (+ r 1)))
          )
      (m (+ v 1) 1))
  )
)
)
(m 100 9)
```

Ejercicio: Que imprime en pantalla el siguiente programa:

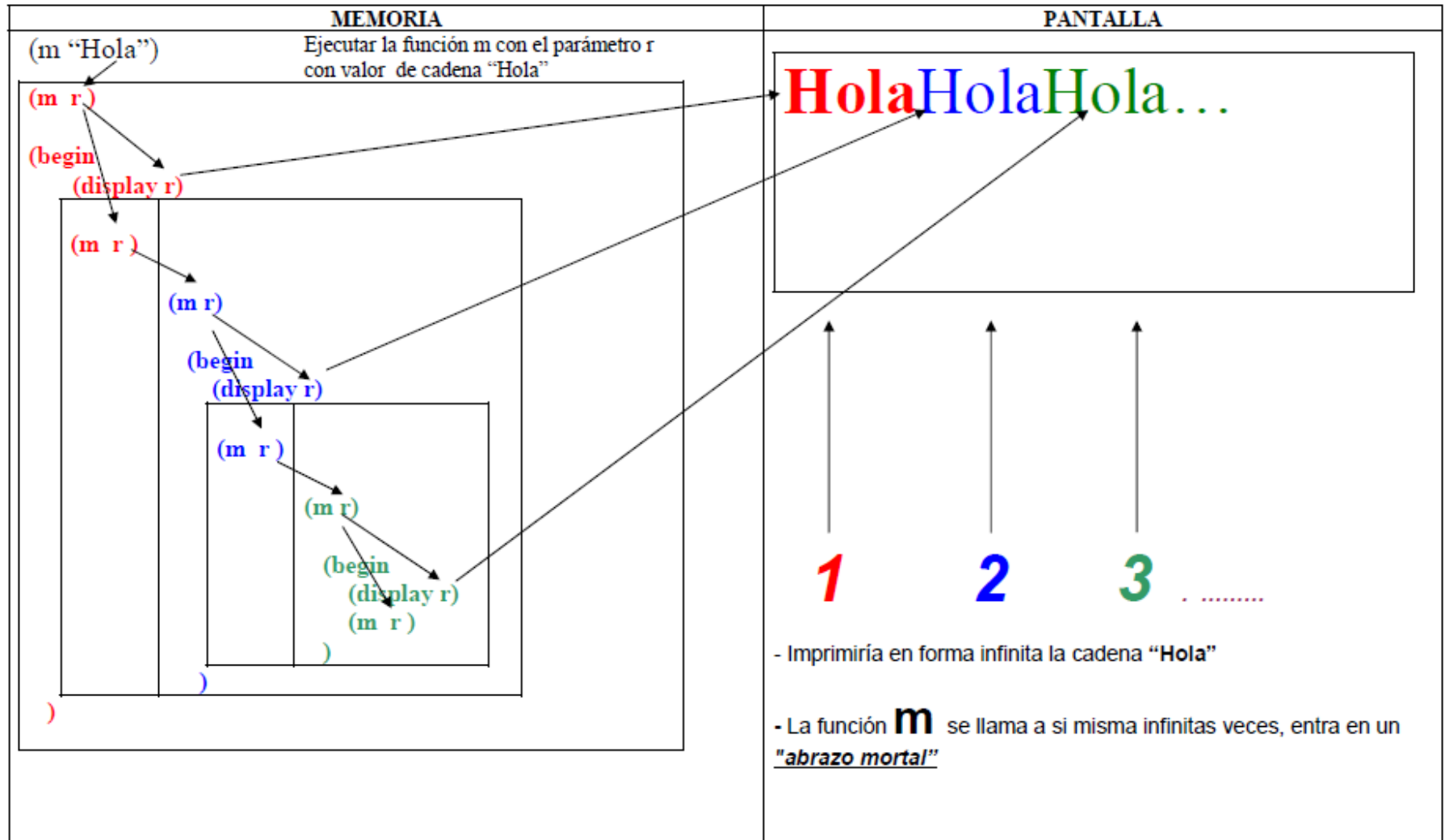
```
(define (m v r)
  (if (<= v 100)
      (if (<= r 10)
          (begin
            (display v) (display " * ") (display r) (display " = ") (display (* v r))
            (newline)
            (m v (+ r 1)))
          )
      (m (+ v 1) 1))
  )
)
)
(m 99 9)
```

Recursividad en Racket

Ejemplo: Dada la siguiente función, hacer la prueba de escritorio

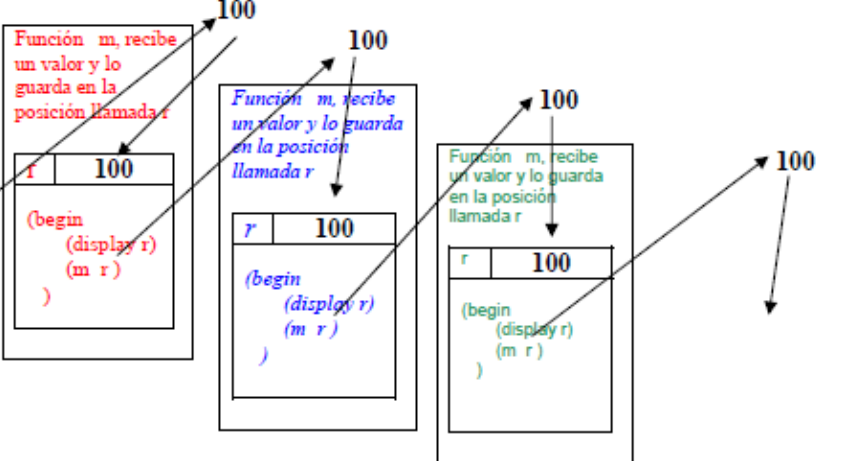
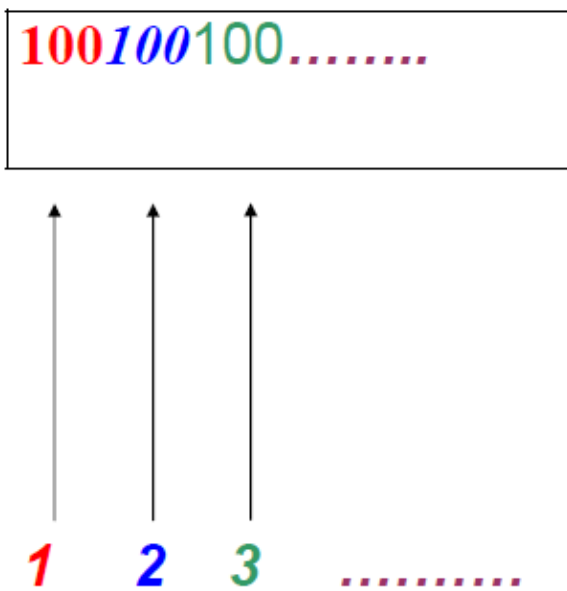
Función	Memoria	Pantalla
<pre>(define (m r) (begin (display r) (m r)))</pre> <p>(m "Hola")</p> <p>LLAMADOS</p>	<p>1 2 3</p>	<p>1 2 3</p> <ul style="list-style-type: none"> - Imprimiría en forma infinita la cadena "Hola" - La función m se llama a si misma infinitas veces, entra en un <u>"abrazo mortal"</u>

Recursividad en Racket

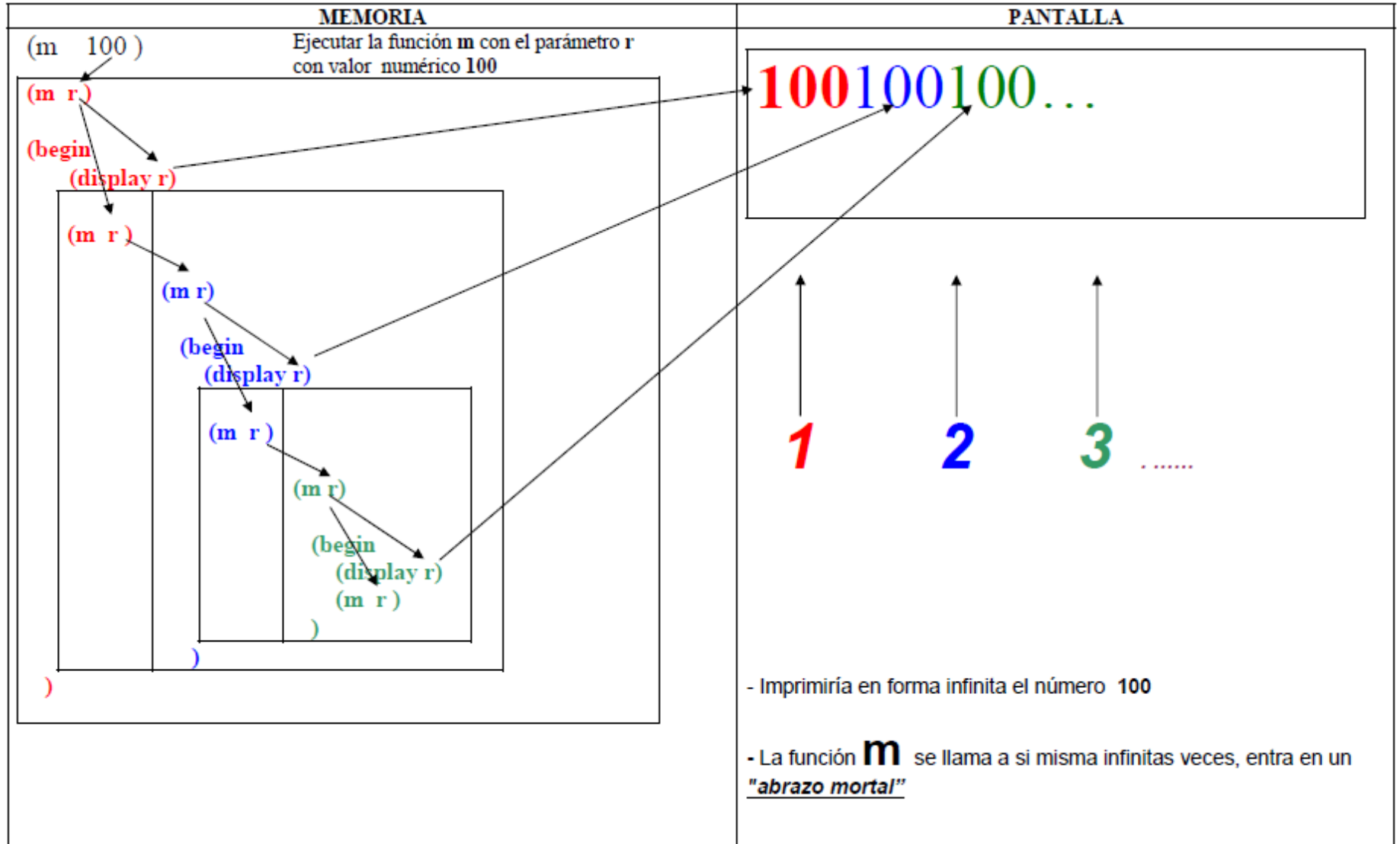


Recursividad en Racket

Ejemplo: Dada la siguiente función, hacer la prueba de escritorio

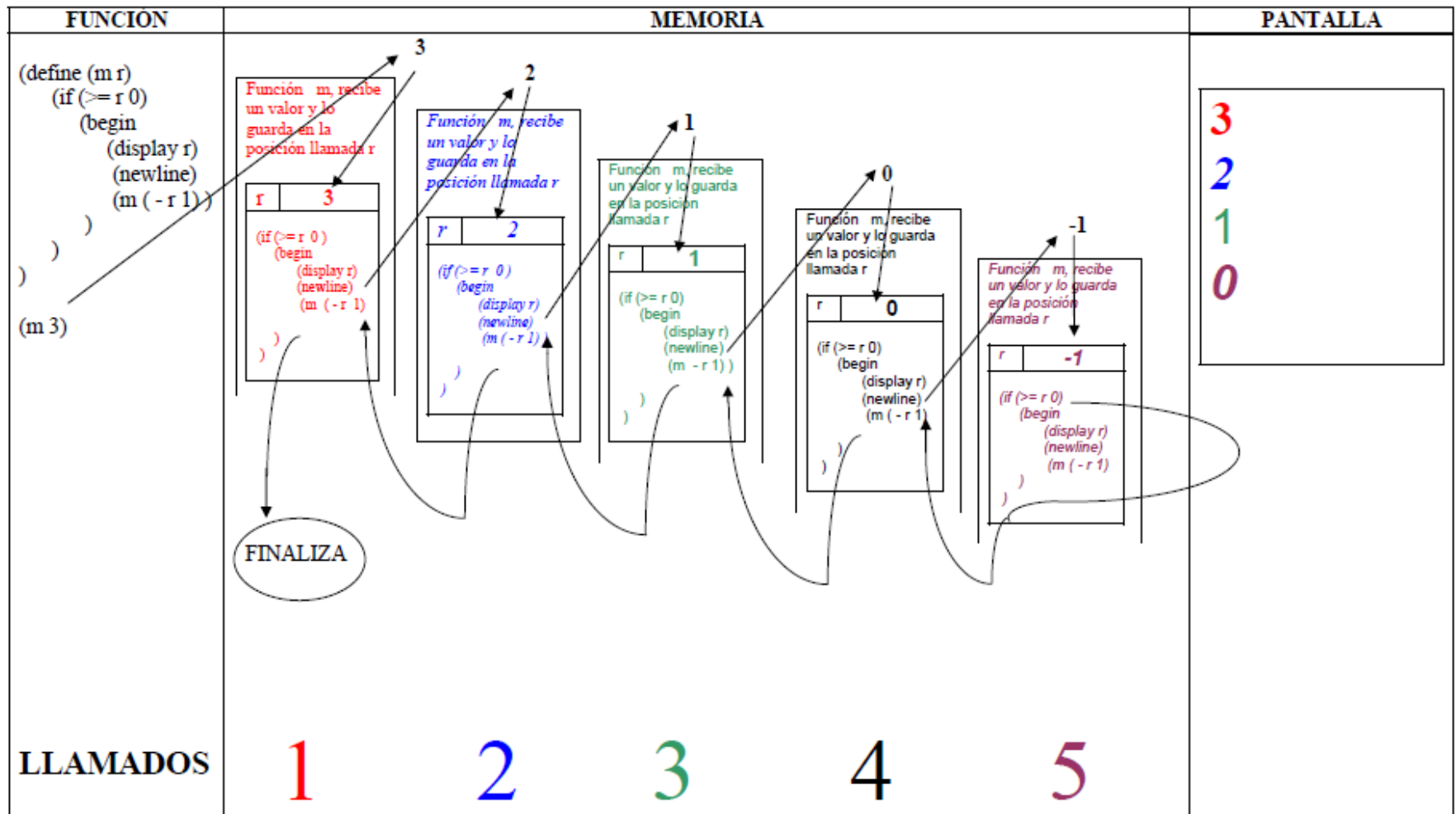
Función	Memoria	Pantalla
<pre>(define (m r) (begin (display r) (m r)))</pre> <p>(m 100)</p> <p>LLAMADOS</p>	 <p>1 2 3 </p>	 <p>- Imprimiría en forma infinita el número 100</p> <p>- La función m se llama a si misma infinitas veces, entra en un "<u>abrazo mortal</u>"</p>

Recursividad en Racket

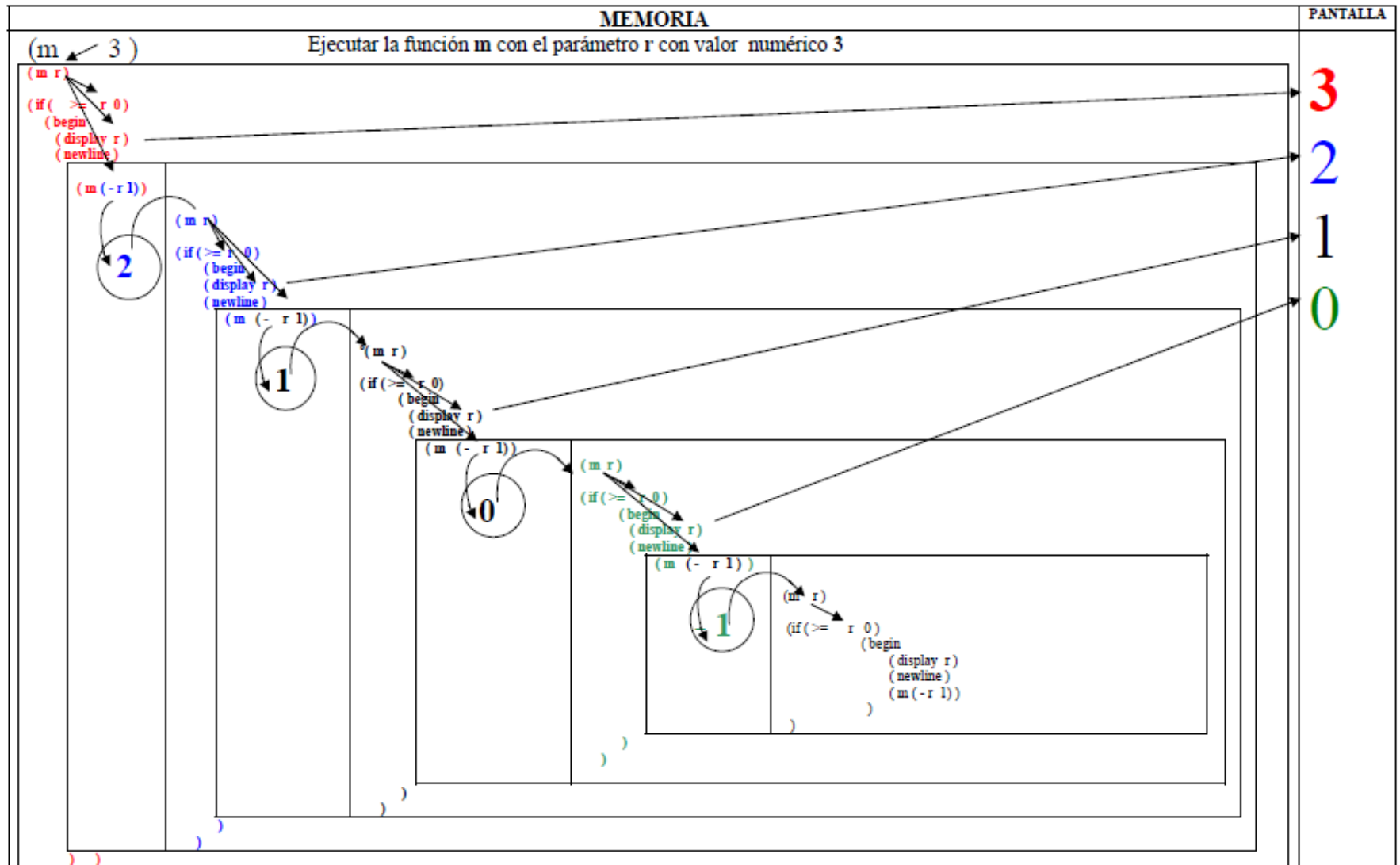


Recursividad en Racket

Ejemplo: Dada la siguiente función, hacer la prueba de escritorio



Recursividad en Racket



Recursividad en Racket

Otra forma de hacer la prueba de escritorio sería:

Función

```
(define (m r)
  (if (>= r 0)
      (begin
        (display r)
        (newline)
        (m (- r 1))
      )
      )
  )
```

(m 3)

MEMORIA							PANTALLA	
Llamado		Primer	Segundo	Tercero	Cuarto	Quinto	<div><div>3</div><div>2</div><div>1</div><div>0</div></div>	
Variables	r	3	2	1	0	-1		

Recursividad en Racket

Ejemplo: Dada la siguiente función, hacer la prueba de escritorio

