

Clase #09 de 27

Introducción a la

Recursividad

Jun 18, Miércoles

Agenda para esta clase

- Revisión Trabajo #2
 - Casteo
 - osfreams
 - SVG
- Introducción a la Recursividad
- Factorial

Revisión Trabajo #2

Casteo

Streams

SVG

Revisión Trabajo #2

- Casteo
- Streams
- SVG

Casteo (!?)

Casteo (Casting)

- Reinterpreta el valor de una expresión como si fuese un valor de otro tipo de dato
- Clásico, menos claro e identificable
 - `(double)3`
 - `(int)3.14`
- Moderno, más claro e identificable
 - `static_cast<double>(3)`
 - `static_cast<int>(3.14)`
 - Otros casteos
 - `const_cast`
 - `dynamic_cast`
 - `reinterpret_cast`
- ¿Cuándo usar Casteo?
 - **NUNCA**
 - Bueno... *caaasi* nunca
 - Es un indicador de que hay que revisar el diseño o modelado de tipo, o que se está haciendo un procesamiento de bajo nivel; de cualquier manera, un casteo requiere verificación de su real necesidad o si hay que rediseñar parte de la solución



Streams No Standard

File Streams

Introducción a Streams No Standard

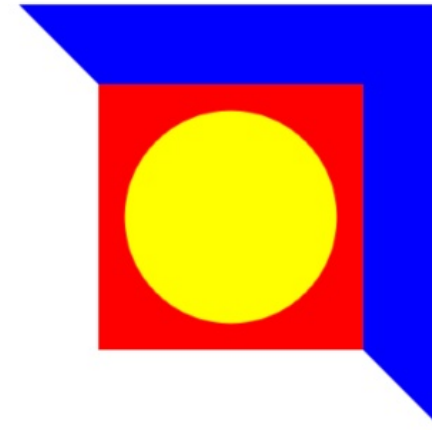
- El ya conocido `cout` es el flujo de salida *estándar*, y su tipo de dato es `ostream`
 - `cout << "Hello";`
 - Como es estándar, siempre está disponible
- También podemos crear flujos de salida *no estándar* que se conecten a archivos con el tipo de dato `std::ofstream` que está en el header `<fstream>`
 - `ofstream out{"output.txt"};`
 - `out << "Hello";`


```
<svg xmlns="http://www.w3.org/2000/svg">  
  <text x="2" y="10">Introducción a SVG</text>  
</svg>
```

Introducción a SVG

```
1 ▼ <svg xmlns="http://www.w3.org/2000/svg" version="1.2" width="100" height="100">
2   <polygon points="10,10 90,10 90,90" fill="blue"/>
3   <rect x="25" y="25" height="50" width="50" fill="red"/>
4   <circle cx="50" cy="50" r="20" fill="yellow" />
5 ▲ </svg>
```

- Es una notación, más precisamente un lenguaje formal, que sirve para describir forma textual gráficos en dos dimensiones (2D)
- Su sintaxis sigue la de XML, y por lo tanto la de HTML
- La notación también permite comentarios
 - `<!-- SVG es simple -->`



Cómo Emitir SVG desde un Programa C++

```
1  #include<iostream>
2  #include<string>
3
4  using std::string;
5  using std::cout;
6
7  void EmitirSvg();
8  void AbrirSvg();
9  void CerrarSvg();
10 void DibujarFiguras();
11 void DibujarPolígono();
12 void DibujarRectángulo(string color);
13 void DibujarCírculo(unsigned radio, string color);
14
15 ▼ int main(){
16     EmitirSvg();
17 ▲ }
18
19 ▼ void EmitirSvg(){
20     AbrirSvg();
21     DibujarFiguras();
22     CerrarSvg();
23 ▲ }
```

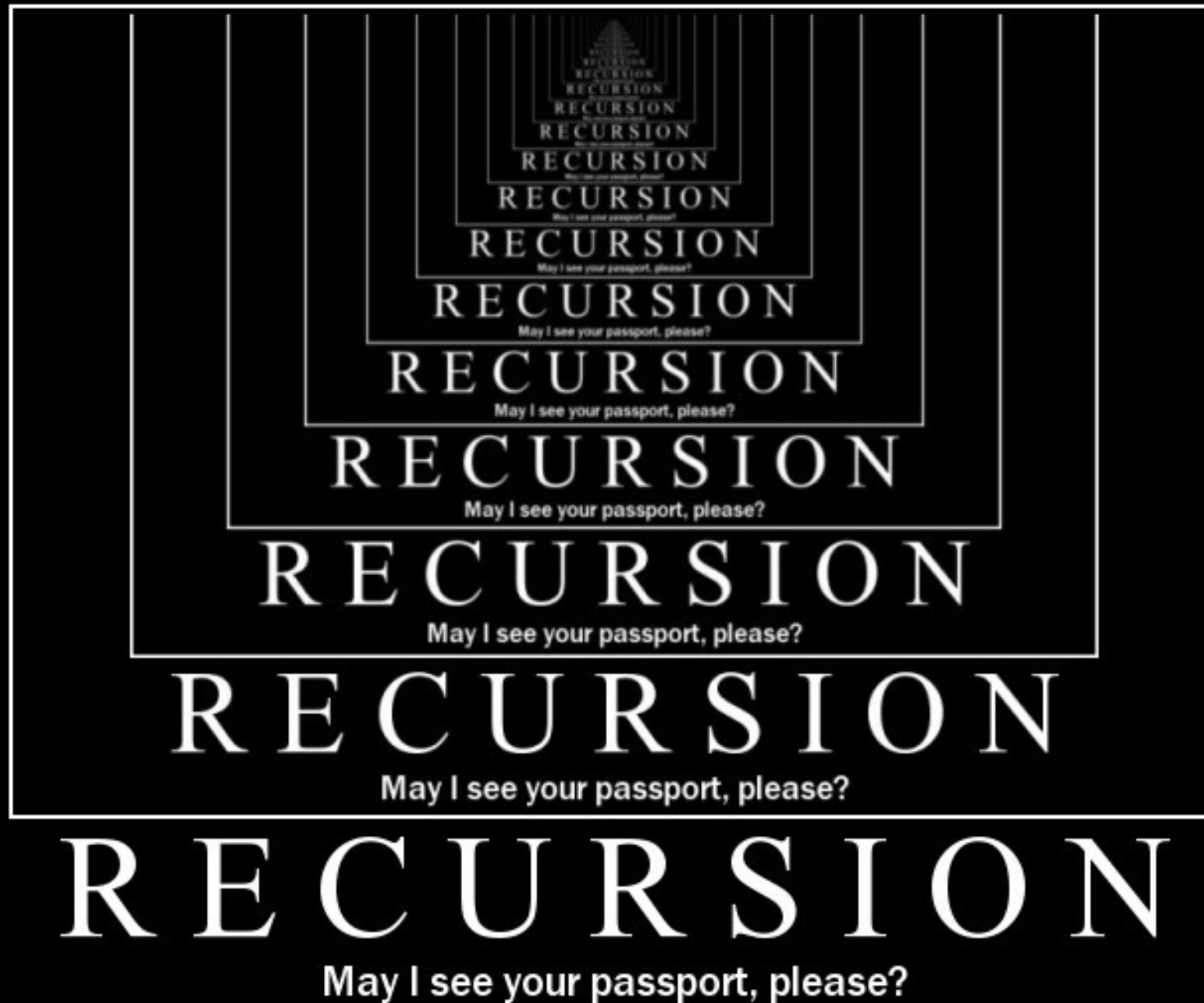
- `"\"Hola\"\\n"s`
- `R__("Hola\\n")s`

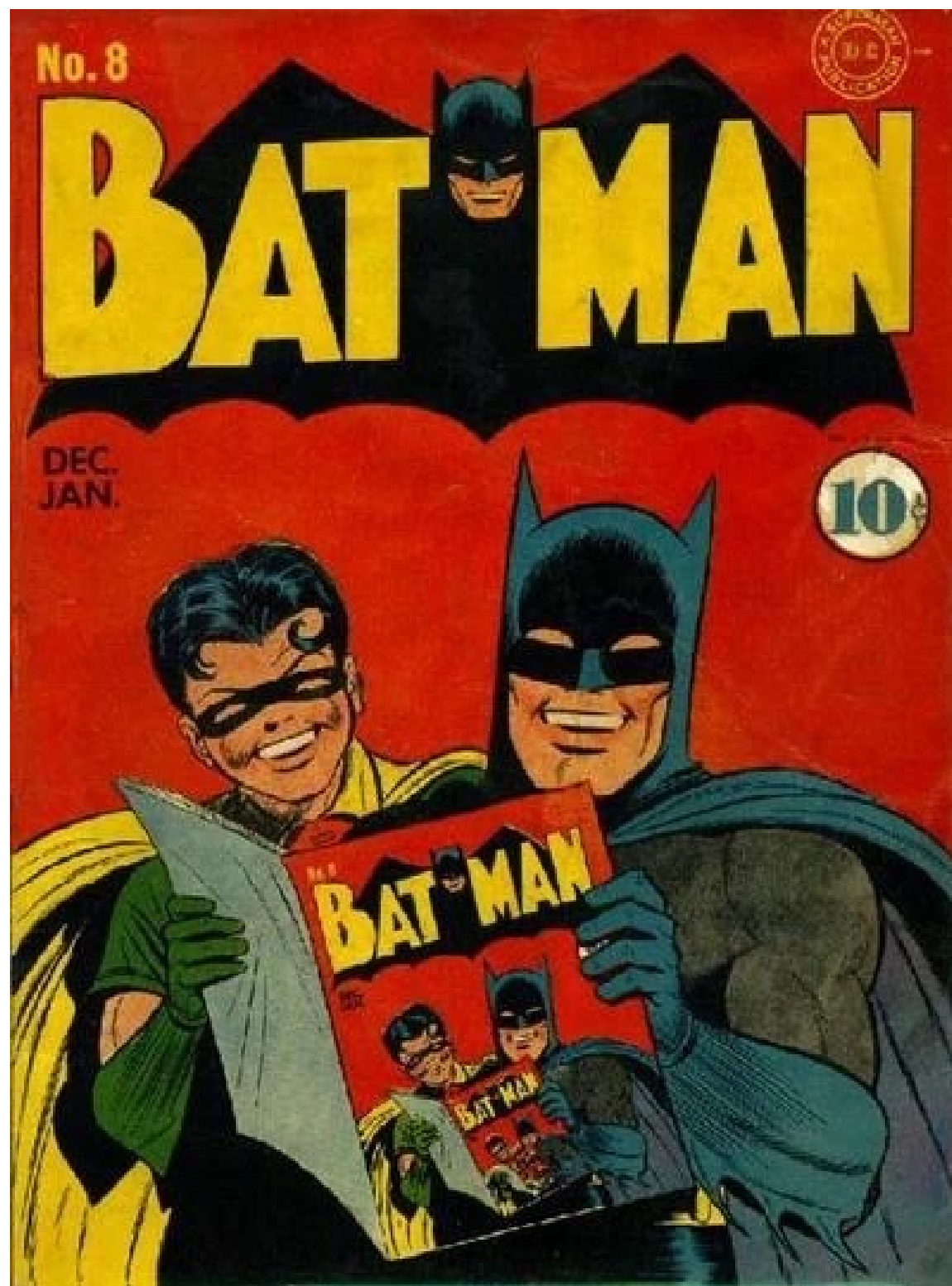
```
33 ▼ void DibujarFiguras(){
34     DibujarPolígono();           // Sin parámetros
35     DibujarRectángulo("red");    // Ejemplo con parámetro color
36     DibujarCírculo(20, "yellow"); // Con radio y color
37 ▲ }
38
39 ▼ void DibujarPolígono(){
40     cout
41         << R"( <polygon points="10,10 90,10 90,90" fill="blue"/>)"
42         << "\\n";
43 ▲ }
44
45 ▼ void DibujarRectángulo(string color){
46     cout
47         << R"( <rect x="25" y="25" height="50" width="50" )"
48         << R"(fill=)" << color << R"( />)"
49         << "\\n"
50         ;
51 ▲ }
52 ▼ void DibujarCírculo(unsigned radio, string color){
53     cout
54         << R"( <circle cx="50" cy="50" )"
55         << R"(r=)" << radio << R"( )"
56         << R"(fill=)" << color << R"( />)"
57         << "\\n"
58         ;
59 ▲ }
```

```
25 ▼ void AbrirSvg(){
26     cout << R("<svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="100" height="100">)" << "\\n";
27 ▲ }
28
29 ▼ void CerrarSvg(){
30     cout << R("</svg>)" << "\\n";
31 ▲ }
```


Introducción a la Recursividad





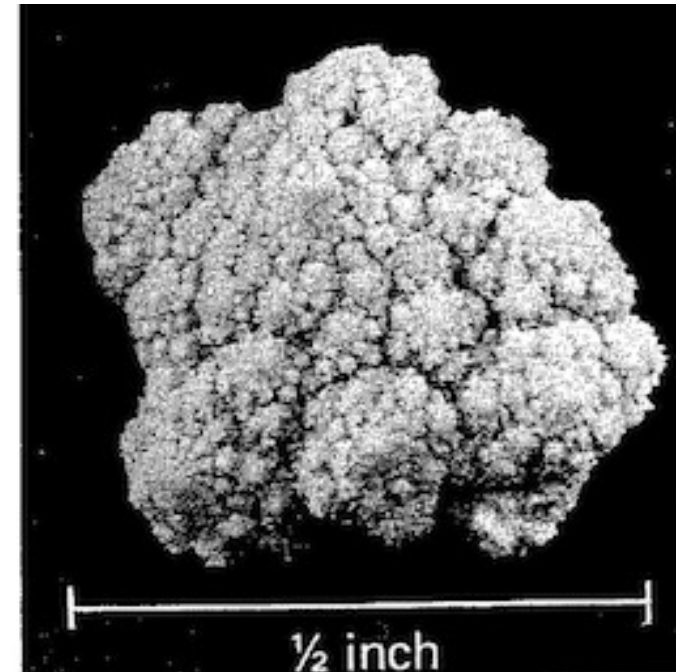
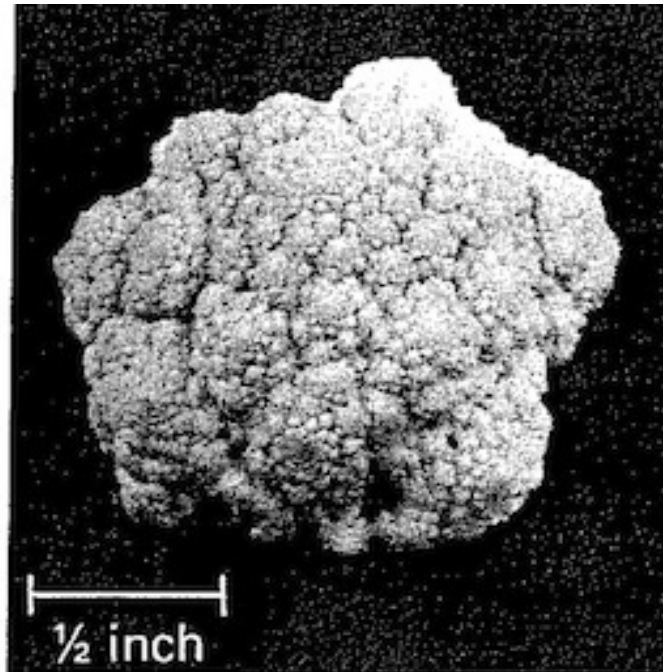
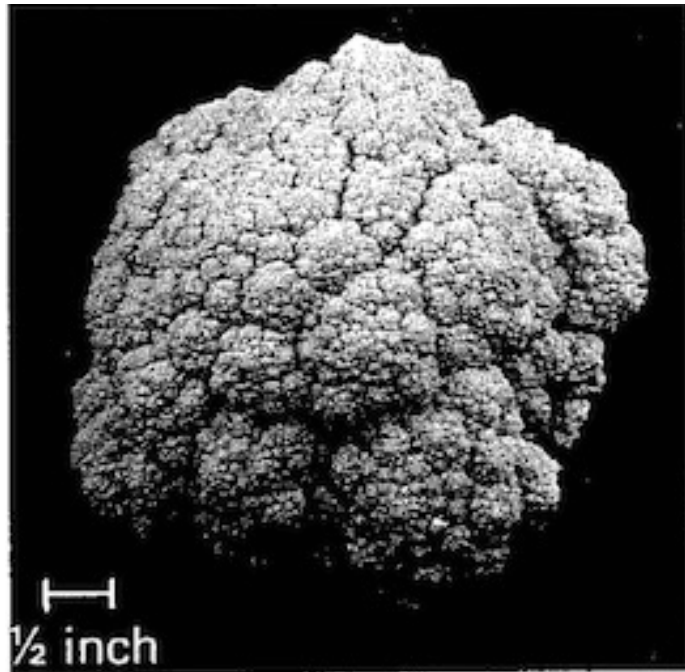


pointer initialization 102, 138
pointer, null 102, 198
pointer subtraction 103, 138, 198
pointer to function 118, 147, 201
pointer to structure 136
pointer, void * 93, 103, 120, 199
pointer vs. array 97, 99–100, 104, 113
pointer-integer conversion 198–199, 205
pointers and subscripts 97, 99, 217
pointers, array of 107
pointers, operations permitted on 103
Polish notation 74
pop function 77
portability 3, 37, 43, 49, 147, 151, 153, 185
position of braces 10
postfix ++ and -- 46, 105
pow library function 24, 251
power function 25, 27
#pragma 233
precedence of operators 17, 52, 95, 131–132, 200
prefix ++ and -- 46, 106
preprocessor, macro 88, 228–233
preprocessor name, __FILE__ 254
preprocessor name, __LINE__ 254
preprocessor names, predefined 233
preprocessor operator, # 90, 230
preprocessor operator, ## 90, 230
preprocessor operator, defined 91, 232

ptrdiff_t type name 103, 147, 206
push function 77
pushback, input 78
putc library function 161, 247
putc macro 176
putchar library function 15, 152, 161, 247
puts library function 164, 247

qsort function 87, 110, 120
qsort library function 253
qualifier, type 208, 211
quicksort 87, 110
quote character, ' 19, 37–38, 193
quote character, " 8, 20, 38, 194

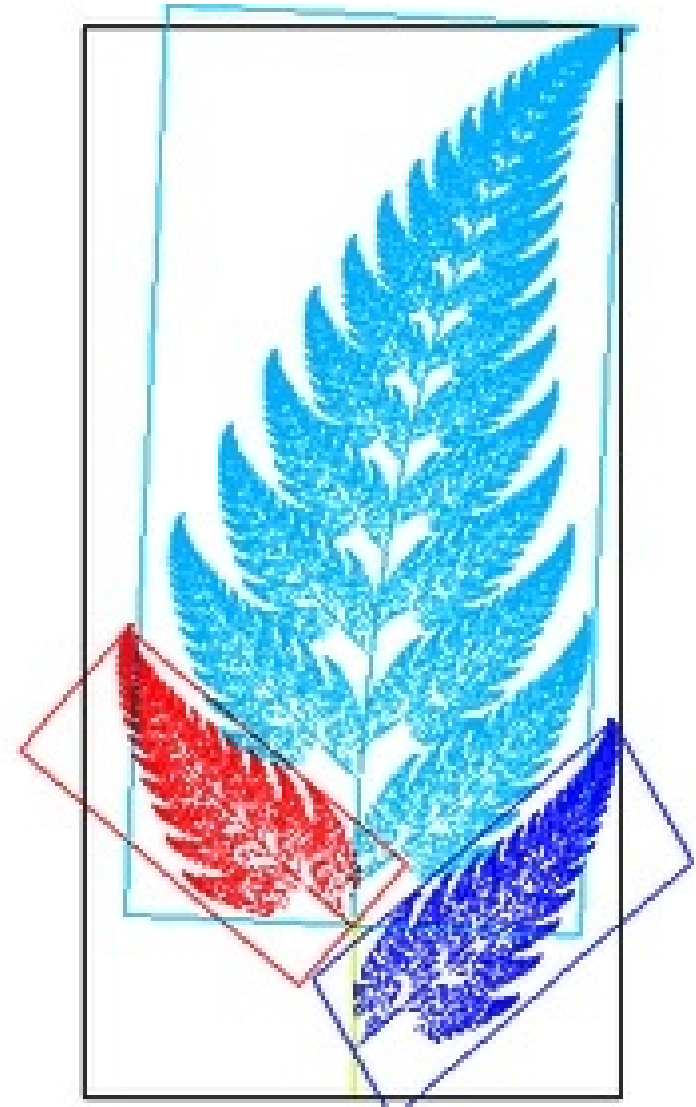
\r carriage return character 38, 193
raise library function 255
rand function 46
rand library function 252
RAND_MAX 252
read system call 170
readdir function 184
readlines function 109
recursion 86, 139, 141, 1 2, 202, 269
redirection see input/output redirection





¿Qué es la Recursividad?

- Es la definición de un proceso basado en su propia definición
- Es la repetición de ítems de una manera *autosimilar*
- *"Para entender recursividad, uno debe entender recursividad"*
- GNU
- Dimensión Temporal o de Proceso
 - Alternativa a la iteración
 - Definiciones de funciones de manera recursiva
- Dimensión Espacial o Estructural
 - Estructuras recursivas



Funciones Recursivas: Factorial

Ejemplo – Factorial: Definición

$$n! = 1 \times 2 \times 3 \times 4 \times \dots \times (n-1) \times n$$

$$n! = \begin{cases} 1 & n = 0 \\ (n-1)! \times n & n > 0 \end{cases}$$

$$\text{Factorial}: \mathbb{N} \rightarrow \mathbb{N} / \text{Factorial}(n) = \begin{cases} 1 & 0 \\ n \cdot \text{Factorial}(n-1) & \text{e.o.c.} \end{cases}$$

$$\text{Factorial}: \mathbb{N} \rightarrow \mathbb{N} / \text{Factorial}(n) = \begin{cases} 1 & n < 2 \\ n \cdot \text{Factorial}(n-1) & \text{e.o.c.} \end{cases}$$

Ejemplo – Factorial: Evaluación

$$\text{Factorial}:\mathbb{N} \rightarrow \mathbb{N} / \text{Factorial}(n) = \begin{cases} 1 & n < 2 \\ n \cdot \text{Factorial}(n - 1) & \text{e.o.c.} \end{cases}$$

$$\text{Factorial}(4) =$$

$$4 \cdot \text{Factorial}(4 - 1) =$$

$$4 \cdot 3 \cdot \text{Factorial}(3 - 1) =$$

$$4 \cdot 3 \cdot 2 \cdot \text{Factorial}(2 - 1) =$$

$$4 \cdot 3 \cdot 2 \cdot \text{Factorial}(1) =$$

$$4 \cdot 3 \cdot 2 \cdot 1 =$$

$$4 \cdot 3 \cdot 2 =$$

$$4 \cdot 6 =$$

$$24$$

Ejemplo – Factorial: Pruebas e Implementación

$$\text{Factorial: } \mathbb{N} \rightarrow \mathbb{N} / \text{Factorial}(n) = \begin{cases} 1 & n < 2 \\ n \cdot \text{Factorial}(n - 1) & \text{e.o.c.} \end{cases}$$

```
unsigned long Fact(unsigned n){ return
    n < 2 ? 1:
        n * Fact(n-1);
}
```

```
#include <cassert>

unsigned long Fact(unsigned);

int main(){
    assert(1 == Fact( 0) );
    assert(1 == Fact( 1) );
    assert(2 == Fact( 2) );
    assert(6 == Fact( 3) );
    assert(24 == Fact( 4) );
    assert(120 == Fact( 5) );
    assert(720 == Fact( 6) );
    assert(5040 == Fact( 7) );
    assert(40320 == Fact( 8) );
    assert(362880 == Fact( 9) );
    assert(3628800 == Fact(10) );
    assert(39916800 == Fact(11) );
    assert(479001600 == Fact(12) );
    assert(6227020800 == Fact(13) );
    assert(87178291200 == Fact(14) );
    assert(1307674368000 == Fact(15) );
    assert(20922789888000 == Fact(16) );
    assert(355687428096000 == Fact(17) );
    assert(6402373705728000 == Fact(18) );
    assert(121645100408832000 == Fact(19) );
    assert(2432902008176640000 == Fact(20) );
}
```

Ejemplo – Factorial: Pila de Invocaciones (I)

Factorial(4)

$4 \cdot \text{Factorial}(4 - 1)$

$4 \cdot 3 \cdot \text{Factorial}(3 - 1)$

$4 \cdot 3 \cdot 2 \cdot \text{Factorial}(2 - 1)$

$4 \cdot 3 \cdot 2 \cdot \text{Factorial}(1)$

$4 \cdot 3 \cdot 2 \cdot 1$

$4 \cdot 3 \cdot 2$

$4 \cdot 6$

24

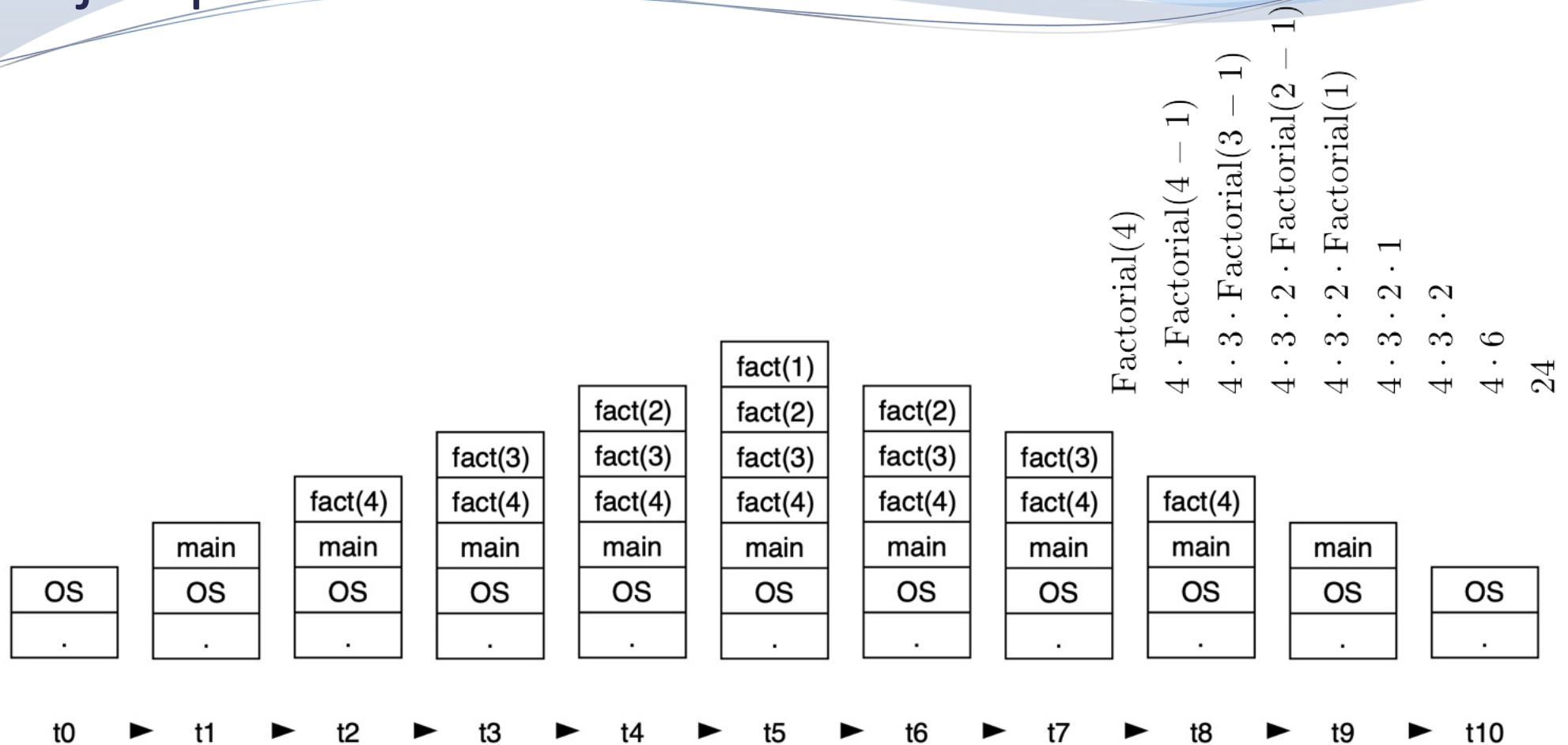
↓₂₄

↓₁

↓₂

↓₆

Ejemplo – Factorial: Pila de Invocaciones



Términos de la clase #09

Definir cada término con la bibliografía

- Casteo
- ofstream
- SVG
- Raw strings
- Introducción a la Recursividad (Recursión)
- Recursividad:
 - Invocación recursiva
 - Caso base
 - Definición de Factorial
 - Implementación recursiva de Factorial
 - Traza de invocaciones
 - Pila de invocaciones

¿Consultas?

Fin de la clase