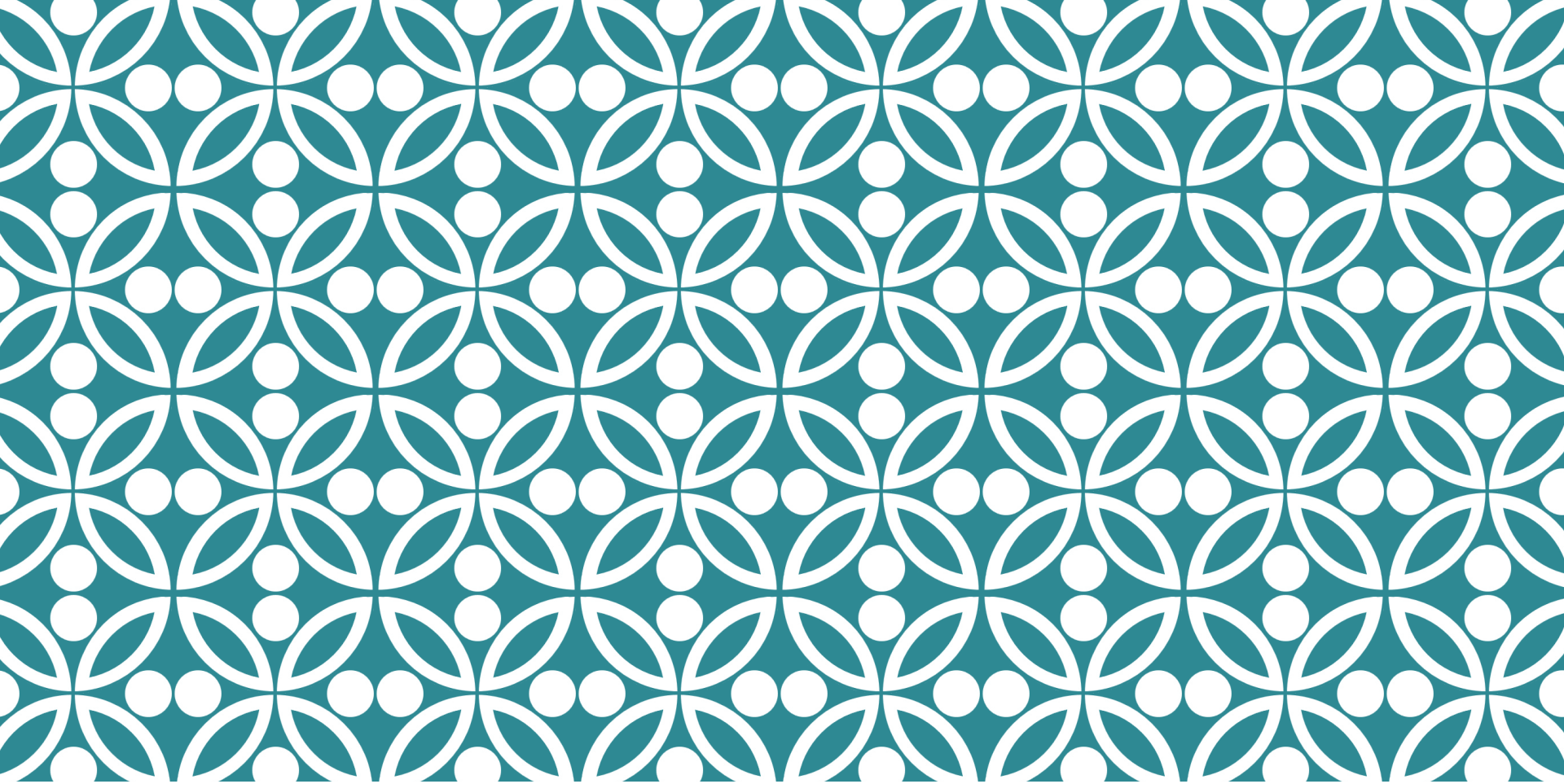

RECURSIVIDAD

Algoritmos y Estructuras de Datos II

Lic. Ana María Company

INTRODUCCIÓN

- A continuación veremos una técnica de programación que tiene su origen en ciertos cálculos matemáticos.
- Esta técnica consiste en describir los cálculos o las acciones de manera auto-alusiva, es decir, resolver problemas describiéndolos en términos de ejemplares más sencillos de sí mismos.

INTRODUCCIÓN

- Se puede entender como un caso particular de la programación con subprogramas (resolución de un problema en términos de otros sub-problemas más sencillos).
- El caso que nos ocupa en esta oportunidad es aquel en el que al menos uno de los sub-problemas es una instancia del problema original.

CONCEPTOS BÁSICOS DE RECURSIÓN

- Los subprogramas recursivos se caracterizan por la posibilidad de invocarse a sí mismos.

- Debe existir al menos un valor del parámetro sobre el que se hace la recursión, llamado caso base, que no provoca un nuevo cálculo recursivo, con lo que finaliza y puede obtenerse la solución.

- En las sucesivas llamadas recursivas los argumentos deben aproximarse a los casos base, para que el proceso concluya al alcanzarse este caso. De lo contrario, se produce la llamada *“recursión infinita"*.

EJEMPLO DE RECURSIÓN

- El Factorial de un número entero positivo n que se define de la siguiente manera:

  $$n! = n * (n - 1) * (n - 2) * \ldots * 1$$

  Como, a su vez: $(n - 1)! = (n - 1) * (n - 2) * \ldots * 1$

- Entonces

  $n!$ se puede definir en términos de $(n - 1)!$ para $n > 0$:

  $$n! = n * (n - 1)!$$

EJEMPLO DE RECURSIÓN

- Por lo tanto, si n <> 0 tendremos que calcular el factorial de n - 1,

  y si es 0 el factorial es directamente 1:

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

- Podemos ver que en la definición del factorial interviene el propio factorial → Este tipo de definiciones en las que interviene lo definido se llaman recursivas.

EJEMPLO DE RECURSIÓN

1! = 1

2! = 2 x 1 = 2

3! = 3 x 2 x 1 = 3 x 2!= 6

4! = 4 x 3 x 2 x 1 = 4 x 3! = 24

5! = 5 x 4 x 3 x 2 x 1 = 5 x 4! = 120

- Podemos observar que el factorial de cada número incluye el factorial de los números que lo anteceden.
- Una buena práctica es encontrar el factor en el resultado que se repite.

FACTORIAL - CÓDIGO EN C

```
10  int factorial( int pNro ) {
11      if( pNro == 0 ) {
12          return 1;
13      } else {
14          return pNro * factorial( pNro - 1 );
15      }
16  }
```

PILA DE LLAMADAS

- Recordemos que en una pila, cuando se inserta un elemento, se añade al principio de la lista. Cuando se lee un elemento, sólo se lee el elemento superior, y se quita de la lista.
- Entonces, la lista sólo dispone de dos acciones: push (insertar) y pop (eliminar y leer).
- Una computadora utiliza internamente una pila llamada pila de llamadas.

QUÉ SUCEDE CUANDO SE INVOCA A UNA FUNCIÓN?

- Dado el siguiente ejemplo:

```c
#include <stdio.h>
#define MAX_CAD 25

typedef char tString[MAX_CAD];

void saludar(tString);
void preguntarPostre();

int main() {
    saludar("Josefina");
    return 0;
}
```

```c
void saludar(tString pNombre) {
    printf("Hola %s!\n", pNombre);
    preguntarPostre();
}

void preguntarPostre() {
    char respuesta;
    printf("Prefieres una fruta(f) o una torta(t)?: ");
    scanf("%c", &respuesta);
    printf("Marcha una %s!",
            respuesta == 'f' ? "fruta" : "torta");
}
```

QUÉ SUCEDE CUANDO SE INVOCA A UNA FUNCIÓN?

- Cuando se invoca a una función saludar(“Josefina“), en primer lugar, la computadora asigna una celda de memoria para la llamada a la función saludar.
- Cada vez que se realiza una llamada a una función, la PC guarda los valores para todas las variables de esa llamada en la memoria.
- A continuación, se ejecutan las sentencias que se encuentren en la llamada, y se invoca a la función preguntarPostre(). De nuevo, la computadora asigna una celda de memoria para esta llamada a la función.
- La computadora utiliza una pila para estas llamadas.

PILA DE LLAMADAS

- La llamada a la segunda función se añade encima de la primera.
- Se ejecutan las sentencias de la 2da llamada.
- Después se regresa a la función de la cual se invocó la 2da llamada.
- Cuando esto ocurre, la llamada de la cima de la pila es quitada.
- Y en la cima de la pila queda ahora la llamada a la función saludar, lo que significa que el control del programa volvió a esta función.
- Cuando llamó a la función preguntarPostre, la función saludar se completó parcialmente.

PILA DE LLAMADAS

- Entonces, cuando se llama a una función desde otra función, la función de llamada se pausa en un estado parcialmente completado.
- Todos los valores de las variables para esa función todavía están almacenados en la memoria.
- Cuando se termina de ejecutar la función preguntarPostre, se regresa a la función saludar y se continúa donde había quedado.
- Esta pila, utilizada para guardar las variables para múltiples funciones, se llama pila de llamadas.

PILA DE LLAMADAS RECURSIVAS

- Esta misma idea es utilizada en las funciones recursivas, que también usan la pila de llamadas.
- Usar la pila es conveniente, pero tiene un costo: guardar toda esa información puede consumir mucha memoria.
- Cada una de esas llamadas a funciones ocupa memoria y, cuando la pila es demasiado alta, significa que la computadora está guardando información para muchas llamadas a funciones.
- Si esto ocurre una opción es re-escribir el código utilizando repetición.

REFERENCIAS

- Adam Drozdek. Data Structures and Algorithms in C++. Fourth Edition.
- Aditya Y. Bhargava. Grokking Algorithms. An illustrated guide for programmers and other curious people.
- Mark Allen Weiss. Estructuras de Datos y Algoritmos. Editorial: Addison-Wesley Iberoamericana.
- Joyanes Aguilar, Luis. Programación en Pascal. 4ª Edición. Editorial: McGraw-Hill/Interamericana de España, S.A.U.
- Cristóbal Pareja Flores, Manuel Ojeda Aciego, Ángel Andeyro Quesada, Carlos Rossi Jiménez. Algoritmos y Programación en Pascal.
- Joyanes Aguilar, Luis. Fundamentos de la Programación. Algoritmos, Estructuras de Datos y Objetos. 3ª Edición. Editorial: McGraw-Hill.
- Luis Joyanes Aguilar, Ignacio Zahonero Martinez. Programación en C. Metodología, algoritmos y estructura de datos. Editorial: McGraw-Hill.