


# Redictado Taller de Programación 2022

## CLASE 3

### Recursión

A silver laptop sits on a light-colored wooden desk. The laptop screen is white and displays four lines of black text in a monospaced font, which is Pascal code for a simple program that prints 'Hola mundo'.

```
Program HolaMundo;  
Begin  
  writeln('Hola mundo');  
end.
```

# Temas de la clase

- Recursión. Concepto. Motivación
- Ejemplo de recursión
- ¿Cómo funciona la recursión?
- Características de un algoritmo recursivo
- Ejercitación

# Recursión

La recursión es una **metodología** para resolver problemas.

Permite resolver un problema **P** por resolución de instancias más pequeñas **P<sub>1</sub>**, **P<sub>2</sub>**, ..., **P<sub>n</sub>** del mismo problema.

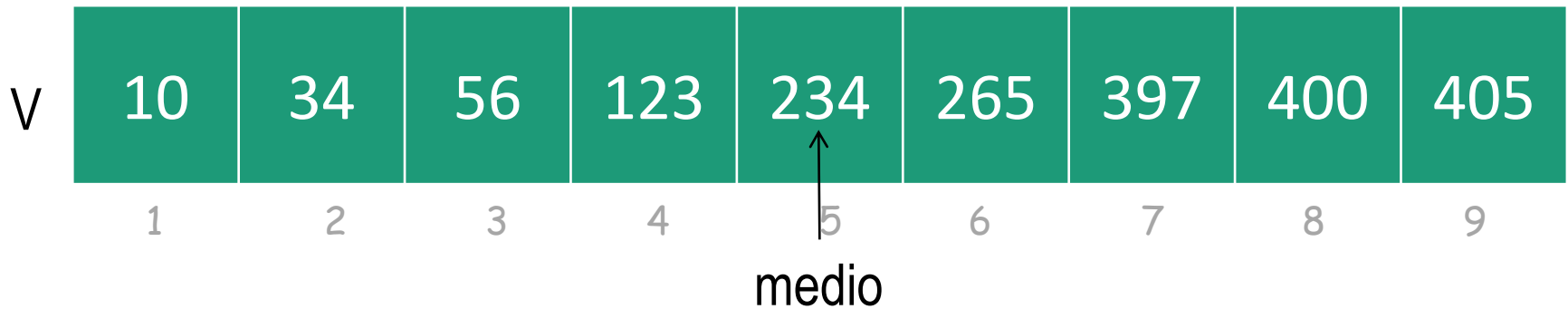
El problema **P<sub>i</sub>** es de la misma naturaleza que el problema original, pero en algún sentido es más simple.

**Veamos un ejemplo ...**

# Recursión

## Ejemplo: Búsqueda dicotómica en un vector

Caso: Se busca en el vector  $V$  el valor 56



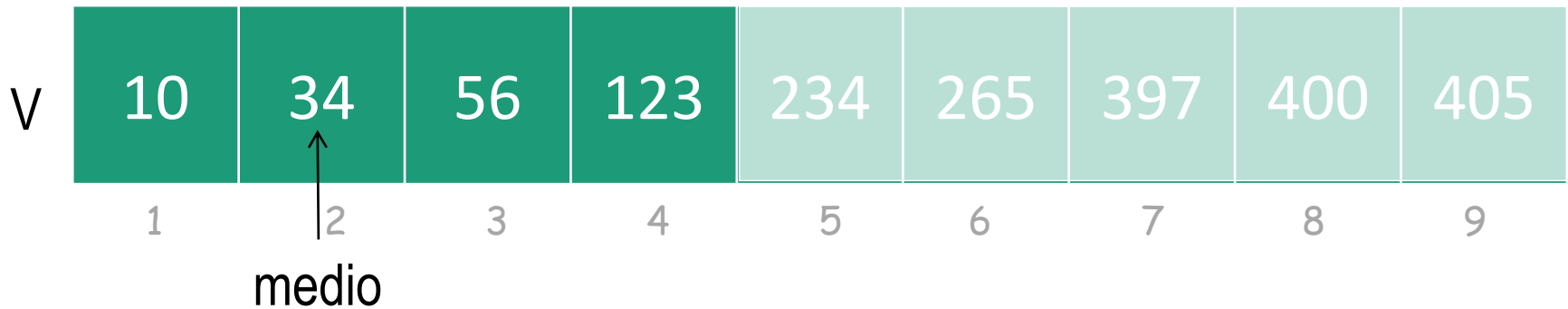
¿Cómo es 56 con respecto a  $v[\text{medio}]$ ?

1. Si es = terminé
2. Si es < busco en la mitad inferior
3. Si es > busco en la mitad superior

# Recursión

## Ejemplo: Búsqueda dicotómica en un vector

Caso: Se busca en el vector  $V$  el valor 56



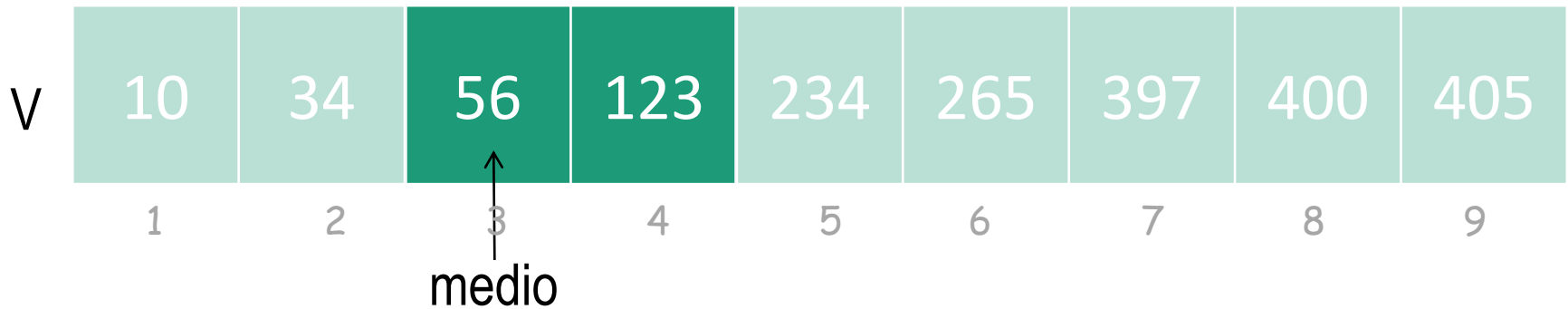
¿Cómo es 56 con respecto a  $v[\text{medio}]$ ?

1. Si es = terminé
2. Si es < busco en la mitad inferior
3. Si es > busco en la mitad superior

# Recursión

## Ejemplo: Búsqueda dicotómica en un vector

Caso: Se busca en el vector  $V$  el valor 56



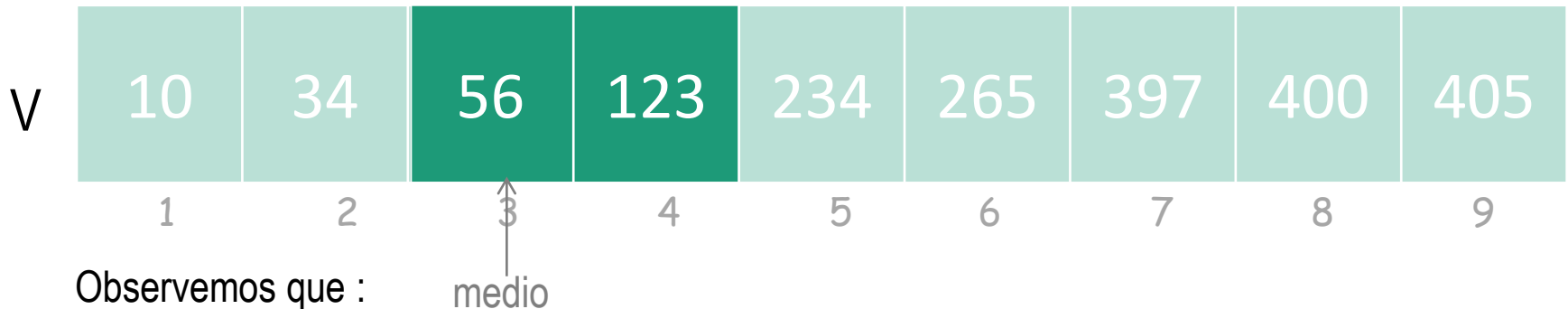
¿Cómo es 56 con respecto a  $v[\text{medio}]$ ?

1. Si es = terminé
2. Si es < busco en la mitad inferior
3. Si es > busco en la mitad superior

# Recursión

## Ejemplo: Búsqueda dicotómica en un vector

Caso: Se busca en el vector V el valor 56



1. La primera vez se trabaja con el vector completo para determinar el punto medio
2. La siguiente vez, el vector se reduce a la mitad
3. La siguiente vez, el vector se reduce a la mitad de la mitad

*¿Cómo se calcula el medio?*

*¿Cómo se calcula la primera mitad?*

*¿Cómo se calcula la segunda mitad?*

*¿Qué pasa si el valor no existía?*

# Recursión

## Ejemplo: Búsqueda dicotómica en un vector (PSEUDOCÓDIGO)

```
Buscar (vector, datoABuscar)
  si el vector “no tiene elementos” entonces
    No lo encontré y termino la búsqueda
  sino
    Determinar el punto medio del vector
    Comparar datoABuscar con el contenido del punto medio
    si coincide entonces
      “Lo encontré”
    sino
      si datoABuscar < contenido del punto medio entonces
        Buscar (1era mitad del vector, datoABuscar)
      sino
        Buscar (2da mitad del vector, datoABuscar)
```



# Recursión

## Ejemplo: Búsqueda dicotómica en un vector

**Buscar** (vector, datoABuscar)

si el vector “no tiene elementos” entonces

No lo encontré y termino la búsqueda

sino

Determinar el punto medio del vector

Comparar **datoABuscar** con el contenido del punto medio

si coincide entonces

“Lo encontré”

sino

si **datoABuscar** < contenido del punto medio entonces

**Buscar** (1era mitad del vector, datoABuscar)

sino

**Buscar** (2da mitad del vector, datoABuscar)

3) Existen 2 casos que se resuelven de manera directa (casos base):

- a) Cuando el vector “no contiene elementos”
- b) Cuando encuentro el datoABuscar

2) En cada llamada, el tamaño del vector se reduce a la mitad.

1) El módulo realiza invocaciones a si mismo

# Recursión

## Características de un algoritmo recursivo

Una **solución recursiva** resuelve un problema por resolución de instancias más pequeñas del mismo problema.

Un algoritmo recursivo involucra:

- al menos una condición de terminación (implícita / explícita)
- al menos una *autoinvocación* (llamada recursiva). Se debe garantizar que en un número finito de *autoinvocaciones* se alcanza la condición de terminación.

# Recursión

## Ejemplo: Potencia de un número

### Planteo de solución recursiva. Tener en cuenta:

1. ¿Cómo defino el problema en términos de problemas más simples del mismo tipo?
2. ¿Cómo achico el problema en cada llamado recursivo?
3. ¿Qué instancia/s del problema son caso/s base?

$$X^n \begin{cases} 1 & \text{si } n = 0 & \text{Caso base} \\ X * X^{n-1} & \text{si } n \geq 1 & \text{Recursión} \end{cases}$$

### Ejemplo de cálculo de $2^3$ :

$$2^3 = 2 * 2^2 = 2 * \underbrace{2 * 2^1}_{2^2} = 2 * 2 * \underbrace{2 * 2^0}_{2^1} = 2 * 2 * 2 * \underbrace{1}_{2^0} = 8$$

# Recursión

## Ejemplo: Potencia de un número

$$X^n \begin{cases} 1 & \text{si } n=0 \\ X * X^{n-1} & \text{si } n \geq 1 \end{cases}$$

```
Function potencia (x,n: integer): real;  
begin  
  if (n = 0) then  
    potencia:= 1  
  else  
    potencia := x * potencia(x,n-1);  
end;
```



# Actividades en Máquina

## ACTIVIDAD 1

Crear el programa **CalculoDePotencia.pas**

```
Function potencia (x,n: integer): real;  
begin  
    if (n = 0) then  
        potencia:= 1  
    else  
        potencia := x * potencia(x,n-1);  
    end;
```

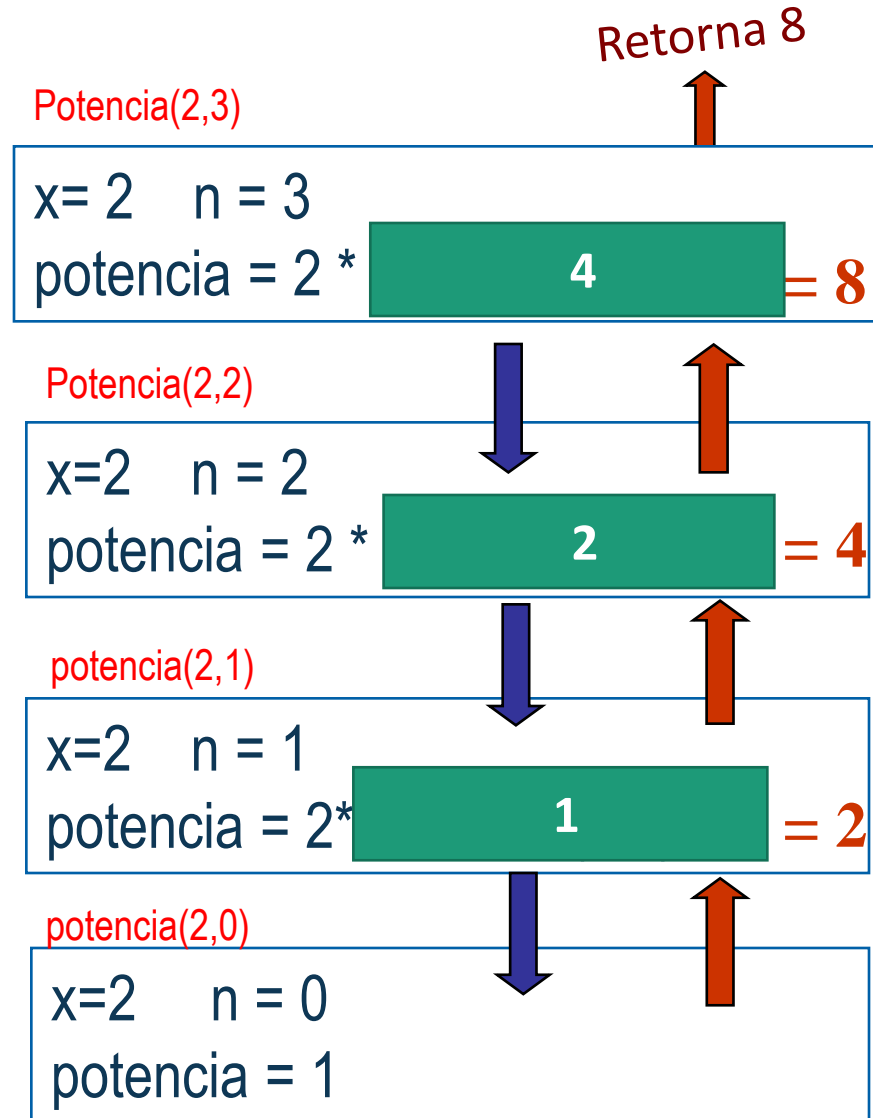
- a) Copiar la función **potencia**.
- b) Completar el programa **CalculoDePotencia** para que lea dos valores  $X$  y  $n$ , invoque a la función **potencia** para calcular  $X^n$  y muestre el resultado.

**Veamos cómo funciona la recursión ...**

# Recursión

## Ejemplo: Potencia de un número

```
program ejemplo;  
  
function potencia (x,n:integer): real;  
begin  
    if (n = 0) then  
        potencia:= 1  
    else  
        potencia := x * potencia(x,n-1);  
    end;  
  
var  
    x,n:integer;  
begin  
    read (x,n);  
    write(potencia(x,n));  
end.
```





# Actividades en Máquina

## ACTIVIDAD 2

a) Copiar en el programa **CalculoDePotencia** la función **potencia1**

```
Function potencia1 (x,n: integer): real;  
begin  
    potencia1 := x * potencia1(x,n-1);  
end;
```

b) Invocar a la función **potencia1** para calcular  $5^3$  .

c) Compilar y ejecutar. ¿Qué ocurre? ¿Por qué?



# Actividades en Máquina

## ACTIVIDAD 3

a) Implementar en el programa **CalculoDePotencia** la función **potencia2**

```
Function potencia2 (x,n: integer): real;  
begin  
  if (n = 0) then  
    potencia2:= 1  
  else  
    potencia2 := x * potencia2(x,n);  
end;
```

b) Invocar a la función potencia2 para calcular  $5^3$ .

c) Compilar y ejecutar. ¿Qué ocurre? ¿Por qué?



## Recordemos el cálculo del dígito máximo de un número entero

```
program CalculoDigitoMaximo;
type digito=-1..9;

var num: integer;

function digitomaximo (n:integer):digito;
var
    max, dig: digito;

begin
    max:= -1;
    while (n<>0) do begin
        dig:= n mod 10;
        if dig > max then max:= dig;
        n:= n div 10;
    end;
    digitomaximo:= max;
end;

begin
    read (num);
    write ('El digito maximo de ', num, ' es: ', digitomaximo(num));
end.
```

Pensemos una solución recursiva usando un procedure ...

# Cálculo del dígito máximo de un número entero (Sol. Recursiva)

```
program CalculoDigitoMaximoRec;
type digito=-1..9;

var num: integer;
    max: digito;

procedure digitoMaximoRec(n: integer; var max: digito);
var
    dig: integer;
begin
    if (n <> 0) then begin
        dig:= n mod 10;
        if (dig > max) then
            max:= dig;

        n:= n div 10;
        digitoMaximoRec(n, max);
    end;
end;

begin
    read (num);
    max := -1;
    digitoMaximoRec(num, max);
    write ('El digito maximo de ', num, ' es: ', max);
end.
```

```
program CalculoDigitoMaximo;
type digito=-1..9;

var num: integer;

function digitomaximo (n:integer):digito;
var
    max, dig: digito;

begin
    max:= -1;
    while (n<>0) do begin
        dig:= n mod 10;
        if dig > max then max:= dig;
        n:= n div 10;
    end;
    digitomaximo:= max;
end;

begin
    read (num);
    write ('El digito maximo de ', num, ' es: ', digitomaximo(num));
end.
```

**Veamos cómo funciona la recursión ...**

# ¿Cómo funciona?

Prog. ppal

Max = **3** ←  
digitoMaximo(132, max)

digitoMaximo(132, max)

n = **13**

Max ←

*dig = 2*

digitoMaximo(13, max)

n = **1**

Max ←

*dig = 3*

digitoMaximo(1, max)

n = **0**

Max ←

*dig = 1*

digitoMaximo(0, max)

n = 0

Max

```
procedure digitoMaximoRec(n: integer; var max: digito);  
var  
  dig: integer;  
begin  
  if (n <> 0) then begin  
    dig:= n mod 10;  
    if (dig > max) then  
      max:= dig;  
    n:= n div 10;  
    digitoMaximoRec(n, max);  
  end;  
end;
```



# Actividades en Máquina

## ACTIVIDAD 4

Descargar el programa **Recursion.pas**

- a) Repase el procedimiento **digitoMaximoRec**
  - ¿Cuál es el caso base?
  - ¿Cómo se acerca al caso base?
- b) Compile, ejecute y compruebe el resultado.



# Actividades en Máquina

## ACTIVIDAD 5

Utilizando el programa **Recursion.pas** realice las siguientes actividades:

a) Modificar el procedimiento **digitoMaximoRec**. Debe colocarse la instrucción **writeln ('max: ', max);** antes de la invocación al procedimiento.

b) Responder:

¿Qué valor se muestra antes de cada llamada recursiva? ¿Por qué?



# Actividades en Máquina

## ACTIVIDAD 6

Utilizando el programa **Recursion.pas** realice las siguientes actividades:

a) Modificar el procedimiento **digitoMaximoRec**. Debe colocarse la instrucción **writeln ('max: ', max);** después de la invocación al procedimiento.

b) Responder:

- ¿Qué valor se muestra antes de finalizar cada instancia recursiva?
- ¿Qué valor se muestra en el programa principal?
- ¿Qué valores se imprimen si el parámetro max es pasado por valor? ¿Qué imprime en el programa? ¿Funciona?



# Actividades en Máquina

## ACTIVIDAD 7

Si el nro es: 5236

**ImprimirDigitos1**

6

3

2

5

Si el nro es: 5236

**ImprimirDigitos2**

5

2

3

6

En el programa **Recursion.pas**

- a) Implementar el procedimiento recursivo **ImprimirDigitos1** que imprime los dígitos de un número dado, empezando por la unidad.
- b) Implementar el procedimiento recursivo **ImprimirDigitos2** que imprime los dígitos de un número dado, finalizando con la unidad.

Nota: el planteo de la solución es similar a la del procedure digitoMaximoRec



# Actividades en Máquina

## ACTIVIDAD 8

Crear el programa **ListaConRecursion.pas** que:

- a) Genere una lista de números enteros y muestre los valores guardados (utilizar los módulos del **programaLista.pas** ya visto)
- b) Invoque a un módulo recursivo **ImprimirEnOrden** que imprima los valores contenidos en la lista en el orden en que se guardaron.
- c) Invoque a un módulo recursivo **ImprimirOrdenInverso** que imprima los valores contenidos en la lista desde el último dato al primero.





# Actividades en Máquina

## ACTIVIDAD 9

En el programa **ListaConRecursion:**

- a) Implementar un módulo recursivo **Máximo** que devuelva el máximo valor de la lista.
- b) Implementar un módulo recursivo **Mínimo** que devuelva el mínimo valor de la lista.
- c) Implementar un módulo recursivo **Buscar** que devuelva verdadero si un valor determinado se encuentra en la lista o falso en caso contrario.



# Actividades en Máquina

## ACTIVIDAD 10

En el **ProgramaVector.pas** de la clase de ordenación

a) Implementar el módulo **BusquedaDicotomica** utilizando el pseudocódigo ya analizado. Utilice el siguiente encabezado:

**Procedure** busquedaDicotomica( v: vector; ini,fin: integer; dato:integer; **var** pos: integer);

*Nota: El parámetro “pos” debe retornar la posición del dato o -1 si el dato no se encuentra en el vector.*

b) Leer un valor e invocar al módulo BusquedaDicotomica

c) Informar el resultado de la búsqueda

Considere todos los casos durante las pruebas



# Actividades en Máquina

## ACTIVIDAD 11

Crear el programa **VectorConRecursion.pas** que:

- a) Genere un vector de números enteros y muestre los valores guardados (utilizar los módulos del **programaVector.pas** ya visto)
- b) Implementar un módulo recursivo **Máximo** que devuelva el máximo valor del vector.
- c) Implementar un módulo recursivo **Suma** que devuelva la suma de los valores contenidos en el vector.
- d) Invocar los módulos implementados e informar el valor máximo y la suma de los valores del vector.