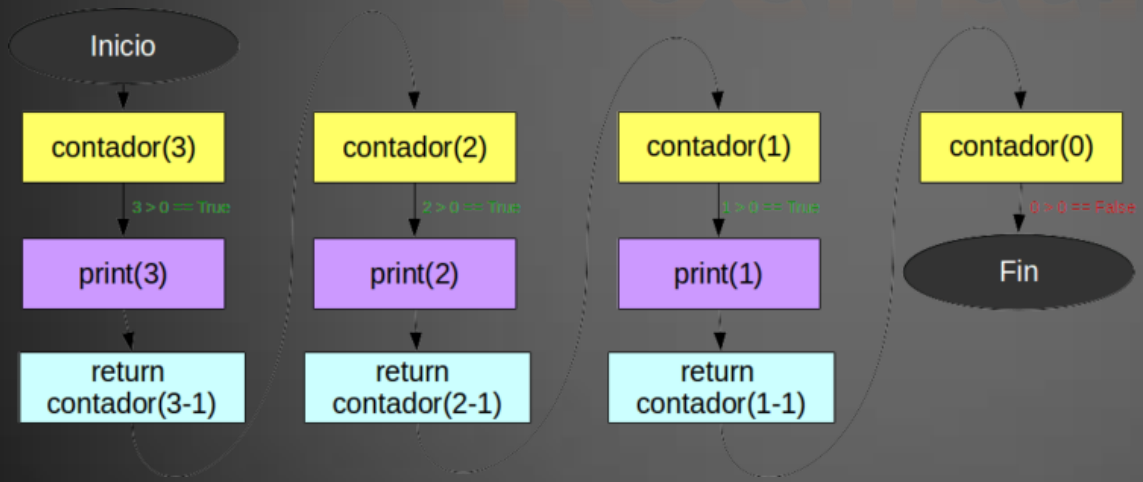
UCAB Universidad Católica ANDRÉS BELLO

# Recursividad




Utilidad de la Recursividad en la construcción de Algoritmos

# Que es la Recursividad

Es un proceso que se define en términos de si mismo, es decir una función que se llama a si misma de forma directa o indirecta. Lo que implica tener:

-) Un razonamiento, lógica o forma de pensar muy matemático.

-) Dificultad para explicar cualquier Proceso utilizando ese mismo Proceso para su definición.

Ejemplo: Cantidad de niños en 1er Grado Iterativamente

Total de Niños en 1er Grado

**Total = 30 + 20 + 25 = 75**

Sección 1


30

+

Sección 2

20

+

Sección 3

25

= 75

# Que es la Recursividad

Expresada esta solución como una solución Iterativa sería:

```
def main( ):
    Secciones=[30, 20, 25]   # Arreglo o Lista en Python

    Total=0
    For Seccion in Secciones :
        Total += Seccion

    Print ("El Total de niños de 1er Grado es: ",Total)

main ( )
```

# Que es la Recursividad

El Planteamiento en forma Recursiva para el mismo ejemplo

Ejemplo: Cantidad de niños en 1er Grado Recursivamente

**Total Secciones**

30

Sección 1

30

+

**Total Secciones**

20

Sección 2

20

+

**Total Secciones**

25

Sección 3

75

25

+

20

+

30

Pila

# Que es la Recursividad

Expresada esta solución como una solución Recursiva sería:

```
def main( ):
    def Total(Secciones) :
        if (len(Secciones)==1) :
            return Secciones[0]

        return Secciones[0] + Total( Secciones[1:])

    Secciones=[30, 20, 25]   # Arreglo o Lista en Python

    x=Total(Secciones)

    Print ("El Total de niños de 1er Grado es: ",x)

main ( )
```

# Que es la Recursividad

- La recursividad es una técnica de programación que consiste en construir una Función que se llame a si misma dentro de la función, de allí se deriva su nombre.

  **Suma(n) = n + Suma(n-1) + Suma (n-2) ……. hasta n = 0 )**

- La recursividad es una forma de implementar un ciclo iterativo a través de Funciones

```
Función  XXX ( n:entero )

    Si ( n = 0 ) entonces
       XXX←1;
       SINO
           XXX← n +XXX ( n - 1 ) ;
    FSi
FFunción
```

- Resolver un problema mediante recursividad significa que la solución depende de las soluciones de pequeñas instancias del mismo problema

  Sumar(2+4+8+2) = ( 2 + Sumar (4+8+2) ) =
  ( 2 + 4 + Sumar(8+2) ) =
  ( 2 + 4 + 8 + Sumar (2) ) = ( 2 + 4 + 8 + 2 ) = 16

- Simplifica el Algoritmo o Código del Programa

# Formas de Implementarla

- Se construyen a través de Funciones

- Las claves para construir una Función Recursiva son:

-) Cada llamada recursiva se debería definir sobre un problema de menor complejidad (algo más fácil de resolver). TOP-DOWN .

-) Ha de existir al menos un **caso base** para evitar que la recursividad no se controle.( Overflow )

- En general los Algoritmos recursivos son más ineficientes en tiempo que los iterativos, aunque suelen tener mucho menos líneas de código y son los únicos y más apropiados para ser utilizados en ciertas Estructuras de Datos.

# Orden en la invocación de una función

**Función 1 ( primero escribir y luego llamar a la función )**

```
Funcion recursiveFunction (num:entero) : entero;
    Si (num < 5) entonces
        Escribir ( num ) ;
        recursiveFunction ← recursiveFunction (num + 1);
    FSi
ffuncion
```

| recursiveFunction ( 0 ) | | | |
|---|---|---|---|
| 1 | recursiveFunction ( 0 ) | | |
| 2 | Escribir ( 0 ) | | |
| 3 | | recursiveFunction ( 0+1 ) | |
| 4 | | Escribir ( 1 ) | |
| 5 | | | recursiveFunction ( 1+1 ) |
| 6 | | | Escribir ( 2 ) |
| 7 | | | | recursiveFunction ( 2+1 ) |
| 8 | | | | Escribir ( 3 ) |
| 9 | | | | | recursiveFunction ( 3+1 ) |
| 10 | | | | | Escribir ( 4 ) |

# Orden en la invocación de una función

**Función 1 ( primero llamar a la función y luego escribir )**

```
Funcion recursiveFunction (num:entero) : entero;
    Si (num < 5) entonces
        recursiveFunction ←recursiveFunction (num + 1);
        Escribir ( num ) ;
    FSi
ffuncion
```

| recursiveFunction ( 0 ) | | | | | |
|---|---|---|---|---|---|
| 1 | recursiveFunction ( 0 ) | | | | |
| 2 | | recursiveFunction ( 0+1 ) | | | |
| 3 | | | recursiveFunction ( 1+1 ) | | |
| 4 | | | | recursiveFunction ( 2+1 ) | |
| 5 | | | | | recursiveFunction ( 3+1 ) |
| 6 | | | | | Escribir ( 4 ) |
| 7 | | | | Escribir ( 3 ) | |
| 8 | | | Escribir ( 2 ) | | |
| 9 | | Escribir ( 1 ) | | | |
| 10 | Escribir ( 0) | | | | |

UCAB Universidad Católica ANDRÉS BELLO

# Recursividad

Ejemplos

# Ejemplos

**1) Se desea elaborar un Algoritmo que a partir de un número N entero positivo MAYOR que uno (1) imprima la cuenta REGRESIVA de ese numero hasta uno ( 1 ) .**

**Iterativamente**

```
Algoritmo
   n,x : entero ;
Inicio
    Escribir(' Indique Nº Mayor que 1' ) ;
    Leer ( n );
    x←n;
    Mientras (x > 0) hacer
        Escribir( x ) ;
        x← x-1;
   FMientras;
Fin
```

**Recursivamente**

```
Algoritmo
   n : entero;
   función Regresiva( x: entero): entero ;
      inicio
         Si (x>0) entonces
            Escribir (x);
            Regresiva←Regresiva (x-1);
          FSi
      fin
Inico
   Escribir('Indique Nº Mayor que 1' ) ;
   Leer ( n );
   Regresiva ( n ) ;
Fin
```

# Ejemplos

**2) Se desea elaborar un Algoritmo que a partir de un número N entero positivo mayor que uno (1) SUME todos los números desde N hasta 1 incluyéndolo.**

.

**Iterativamente**

```
Algoritmo
 n,x ,sum: entero ;
Inicio
   Escribir(‘ Indique Nº  Mayor que 1)
   Leer ( n );
   x←n; sum ← 0;
   Mientras (x>0) hacer
        sum ← sum + x ;
        x← x-1;
   FMientras
   Escribir (‘ La Suma es :  ‘, sum );
Fin
```

**Recursivamente**

```
Algoritmo
     n,sum : entero;
    función Sumar ( x: entero):: entero ;
          Si (x>0) entonces

                Sumar← ( x + Sumar (x-1) ) ;

          FSi
   ffuncion
Inico
   Escribir(‘Indique Nº   para hacer la Sumar)
   Leer ( n );
    sum←  Sumar ( n ) ;
    Escribir ( ‘ La suma  es  :  ‘ , sum) ;
Fin
```

# Ejemplos

**3) Se desea elaborar un Algoritmo que a partir de un número N entero positivo mayor que dos (2) Calcule el FACTORIAL de N .**

.

**Iterativamente**

```
Algoritmo
 n,x ,fact: entero ;
Inicio
   Escribir(‘ Indique Nº  Mayor que 2 para
                calcular su Factorial)
   Leer ( n );
   x←n; fact ← 1;
   Mientras (x>0) hacer
        fact ← fact * x ;
        x← x-1;
   Fmientras;
   Escribir (‘ El Factorial es :  ‘, fact );
Fin
```

**Recursivamente**

```
Algoritmo
     n,sum : entero;
    función Factorial ( x: entero): entero ;
          Si (x>1) entonces

                Factorial← x * Factorial (x-1);

             SINO

                  Factorial ← 1;
          FSi
   ffuncion
Inico
   Escribir(‘Indique Nº  Mayor que 2 para
                 calcular su Factorial);
   Leer ( n );
    fact←  Factorial ( n ) ;
    Escribir ( ‘ El Factorial es  :  ‘ , fact) ;
Fin
```

# Ejemplos

**4) Se desea elaborar un Algoritmo que calcule el Producto de dos números N1 y N2 a través del Método de Sumas Sucesivas.**

**Iterativamente**

```
Algoritmo
 n1,n2, prod,cont: entero ;
Inicio
   Escribir(' Indique dos Nº positivos para
                Multiplicarlos) ;
   Leer ( n1 );  Leer(n2);
   cont←0; prod ← 0;
   Mientras (cont<n2) hacer
      prod ←( prod + n1) ;
      cont← (cont+1);
   FMientras

   Escribir (' El Producto es :  ', prod );

Fin
```

**Recursivamente**

```
Algoritmo
     n1,n2,prod,cont : entero;
    función Producto ( x,c: entero):: entero ;
          Si ( c < n2 ) entonces

                Producto← x + Producto( x, c+1 );

             SINO

                  Producto ← 0;
          FSi
    ffuncion
Inico
    Escribir(Indique dos Nº  positivo para
                  Multiplicarlos)
     Leer ( n1 ) ;  Leer ( n2 ) ; cont←0;
     prod←  Producto ( n1, cont ) ;
     Escribir ( ' El Producto  es  :  ' , prod ) ;
Fin
```

# Ejemplos

**5) Se desea elaborar un Algoritmo que determine la cantidad de dígitos de un número N entero positivo .**

**Iterativamente**

```
Algoritmo
    variables
        n,x, cant: entero;
Inicio
     Escribir ( ' Indique Nº  entero positivo ' ) ;
     Leer ( n ) ;
     x ← n ;
     cant ← 1;
     Mientras ( x div 10 ) < > 0 hacer
          cant ← cant + 1;
          x ←( x div 10 );
     FMientras
    Escribir ( ' La cntidad de digitos es : ',cant)
Fin
```

**Recursivamente**

```
Algoritmo
   variables
       n,c : entero ;
    funcion Cantidad ( x, c : entero ) : entero ;
       Si (x div 10)<>0 entonces

            Cantidad←c+Cantidad((x div 10),1);

            SINO

                Cantidad ← 1;
        Fsi
    ffuncion
Inicio
     Escribir ( ' Indique Nº  entero positivo ' ) ;
     Leer ( n ) ;
      c ← Cantidad ( n, 1 ) ;
     Escribir (' La cantidad de digitos es : ', c ) ;
Fin
```