Recursión

La recursividad consiste en realizar una definición de un concepto en términos del propio concepto que se está definiendo.

Ejemplos:

- El factorial de un número natural n, es 1 si dicho número es 0, o n multiplicado por el factorial del número n-1, en caso contrario.
- La sucesión de Fibonacci
- La n-ésima potencia de un número x, es 1 si n es igual a 0, o el producto de x por la potencia (n-1)-ésima de x, cuando n es mayor que 0.

Recursión

La solución para problemas recursivos puede plantearse de la siguiente forma:

- División sucesiva del problema original en uno o varios más pequeños del mismo tipo.
- Se resuelven los problemas más sencillos.
- Con las soluciones de éstos se construyen las soluciones de los problemas más complejos.

Ejemplo (1)

Cálculo del factorial de un número, por ejemplo, 5.

5! = 5 * 4!

4! = 4 * 3!

3!= 3 * 2! DESCOMPOSICIÓN DEL PROBLEMA

2! = 2 * 1!

1! = 1 * 0!

SOLUCIÓN CONOCIDA O DIRECTA **0! = 1**

1! = 1 * 0! = 1

2! = 2 * 1! = 2

3! = 3 * 2! = 6

4! = 4 * 3! = 24

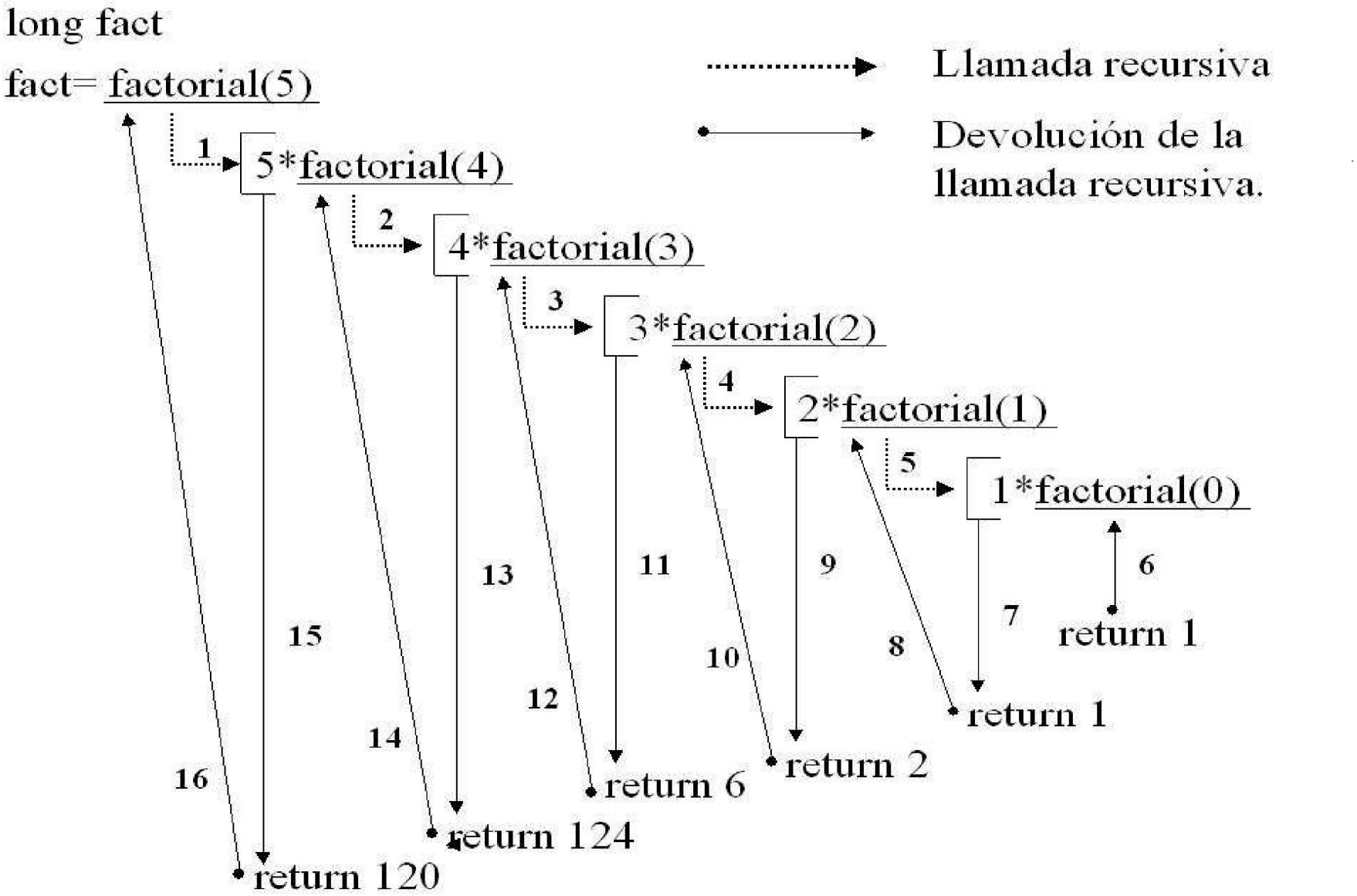
5! = 5 * 4! = 120

El código recursivo sería:

```
factorial (n)
{ if (n = 0) return 1;
else return n * factorial(n-1);
 }
```

El algoritmo recursivo siempre debe tener:

- La llamada recursiva, que expresa el problema original en términos de otro menor,
- El valor para el cual se conoce una solución no recursiva denominado **caso base**: una instancia del problema cuya solución no requiere de llamadas recursivas.



Cómo se ejecuta el programa?

Fibonacci

La sucesión de números F0 = 1, F1 = 1, F2 = 2, F3 = 3, F4 = 5, etc. recibe el nombre de sucesión de Fibonacci.

La expresión recurrente que define los números de Fibonacci es:

Fn = Fn-1 + Fn-2 si n>= 2

F0 = F1 = 1

Este define la secuencia: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, ...

El programa recursivo seria:

```
fibonacci(int n){
    if ((n = 0) or (n = 1)) return 1;
    else return fibonacci (n-1) + fibonacci (n-2);
   }
```

Fibonacci

Se puede analizar y llegar a la conclusión que el algoritmo recursivo toma un tiempo **exponencial**, mientras que por contraste el algoritmo iterativo toma un tiempo **lineal**:

```
fibonacci(int n){
    F[0] = F[1]= 1;
    for (int i=2; i<=n; i++)
        F[i] = F[i-1]+ F[i-2];
    delete n;
```

Ejemplo

Calcular recursivamente la cantidad de veces que ocurre un elemento (objetivo) en una estructura de arreglo (vector)

```
Ocurrencias (n,  vector, objetivo, primero)
{ if (primero > n-1) return 0;
else
{ if (vector[primero] == objetivo)
      return(1+Ocurrencias(n,vector,
           objetivo,primero+1));
else
      return(Ocurrencias(n,vector,
                objetivo, primero+1));
  }
}
```

Cómo se administra la recursión

La memoria a la hora de ejecutar un programa queda dividida en dos partes:

- la zona donde se almacena el código del programa y
- la zona donde se guardan los datos: pila (utilizada para llamadas recursivas).

Cómo se administra la recursión

El programa principal llama a una rutina,

- se crea en la pila de un registro de activación o entorno,
- se almacena constantes, variables locales y parámetros formales.
- estos registros se van apilando conforme se llaman sucesivamente desde una función a otras
- cuando finaliza la ejecución, se va liberando el espacio

A tener en cuenta :

La profundidad de la recursión está dada por el número de registros de activación en la pila en un momento dado.

Problema:

Si profundidad es muy grande => desbordamiento de la pila.

Representación gráfica del registro de activación:

Ejemplo de cómo evoluciona una pila- Ejemplo (3)

Problema: impresión de palabras en sentido contrario al leído.

Función recursiva: imp_OrdenInverso.

Parámetros: número de palabras que quedan por leer (n).

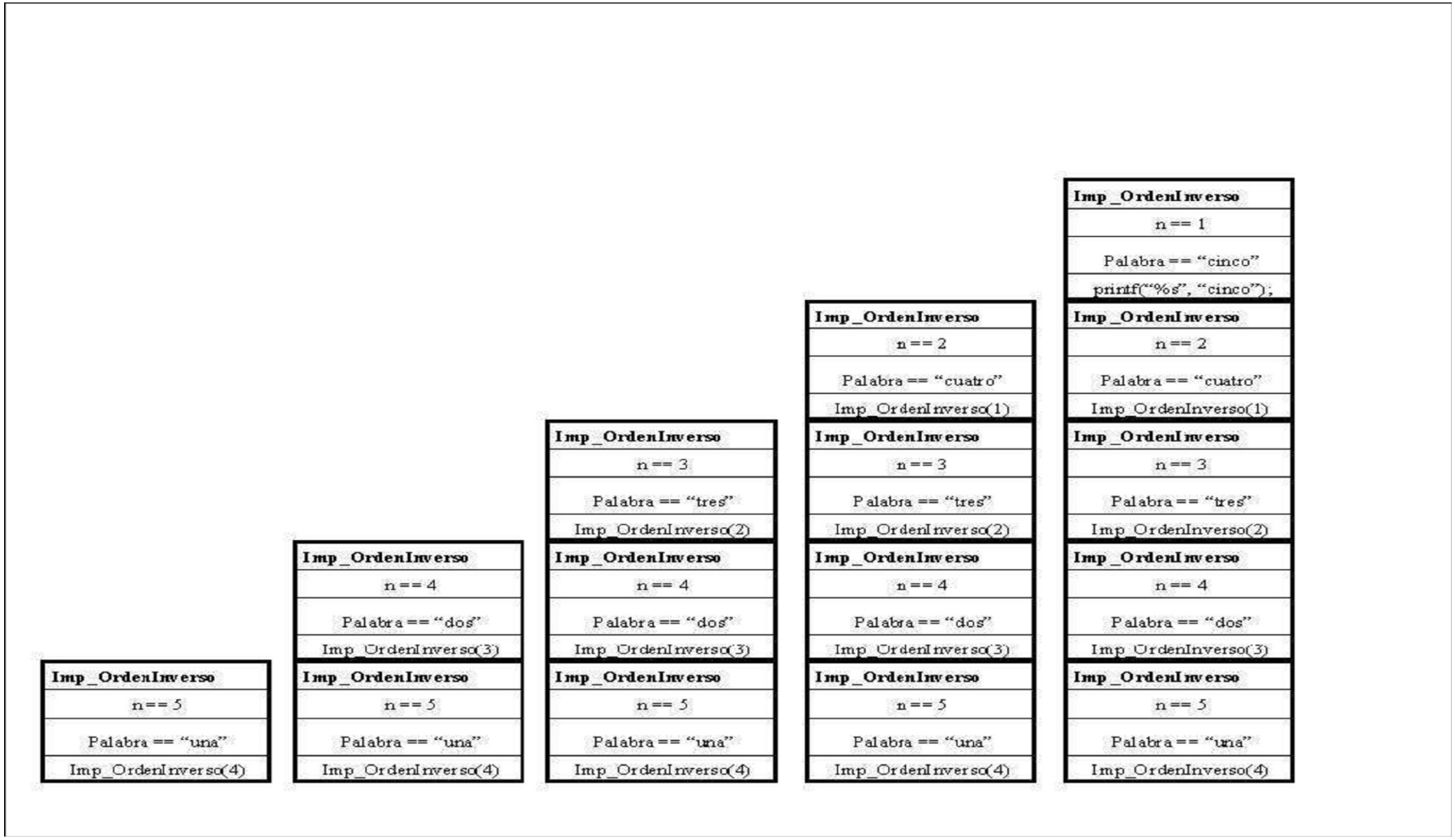
Algoritmo:

- Se lee la primera palabra, y recursivamente se llama a la función para que lea e imprima el resto de palabras, para que finalmente se imprima la primera leída.
  - El caso base: cuando sólo quede una palabra (n == 1), en cuyo caso se imprimirá directamente.
  - Como en cada llamada recursiva hay que leer una palabra menos del total, en n-1 llamadas se habrá alcanzado el caso base.
  - Una vez se haya alcanzado el caso base al devolver continuamente el control a las funciones invocantes se irán imprimiendo de atrás a delante las palabras obtenidas desde la entrada estándar.

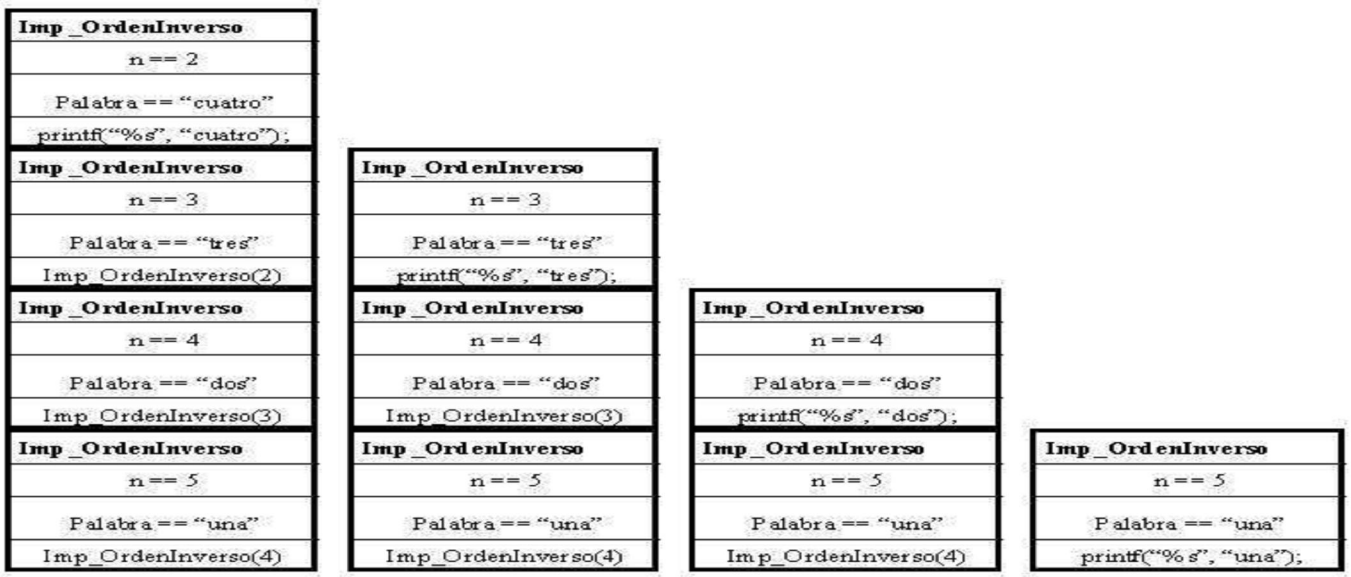
Algoritmo

```
imp_OrdenInverso(int n)
{
 if (n == 1)
{palabra =leerpalabra();
  imprimir(palabra);
}
else
{
   palabra =leerpalabra();
   imp_OrdenInverso(n-1);
   imprimir(palabra);
}
}
```

Administración



Administración



Ejemplo

Dadas dos listas vinculadas, se desea saber si son iguales:

Int iguales {nodo * px , nodo * py}

```
 {if ((px=NULL) && (py =NULL))
        return 1;
     else
      return ((px-> elem == py->elem)
                && iguales (px->sig,py->sig))
  }
```