UNIVERSIDAD NACIONAL DE HURLINGHAM

Estructuras de Datos

Profesores

Ariel Clocchiatti

Sergio Gonzalez

UNIVERSIDAD NACIONAL DE HURLINGHAM

Unidad 2: Recursividad, Arreglos uni y multidimensionales

Profesores

Ariel Clocchiatti

Sergio Gonzalez

Programación modular

- Descomposición en módulos independientes
  - Subprogramas o subalgoritmos
  - Independencia del programa principal
  - Diseño descendente
  - Reutilización de código

Programación modular

- Subprogramas se 'llaman' desde el programa principal
- Abstracción procedimental:
  - Nombre
  - Parámetros de entrada / salida
- Tipos de subprogramas:
  - Funciones
  - Procedimientos / Subrutinas

Programación modular

- Funciones
  - Tareas especificas
  - Lista de valores de entrada (argumentos)
  - Único valor de salida
- Procedimientos / Subrutinas
  - Similar a funciones
  - Pueden devolver ninguno o varios valores

Programación modular

- Recordatorio:
  - Ámbito de las variables (scope)
  - Paso de parametos
    - Por valor
    - Por referencia

Recursividad

- Subprogramas pueden llamarse a si mismos
- Alternativa a la repetición (soluciones iterativas)
- Problemas con ‘estructura de solución recursiva’

Estructura de solución recursiva

- Tenemos un problema A
- El algoritmo diseñado se divide en dos partes B y C
- Si una de esas partes (por ejemplo B) es idéntica a A
- Entonces el problema es recursivo, porque la resolución de B se puede dividir igual que la de A

Estructura de solución recursiva

- En programación, tenemos un procedimiento que se llama a si mismo
- La autollamada resuelve una versión mas chica del problema
- Para que sea finito, debe haber una condición de corte (caso simple sin autollamada)

Método de las 3 preguntas

- Caso base: Existe una salida no recursiva o caso base? Es correcta la solución?
- Mas chico: Cada autollamada es un problema mas chico del original?
- Caso general: Es correcta la solución en los casos no base?

Escritura de programas recursivos

- Obtener definición exacta del problema
- Determinar el tamaño del problema general
- Resolver el/los casos triviales (no recursivos)
- Resolver el caso general en términos de uno mas chico (llamada recursiva)

Ejemplos

- Ejemplo 1: Calculo del factorial
  - N! = N (N - 1)!
  - (N - 1)! = (N – 1) (N - 2)!

Ejemplos

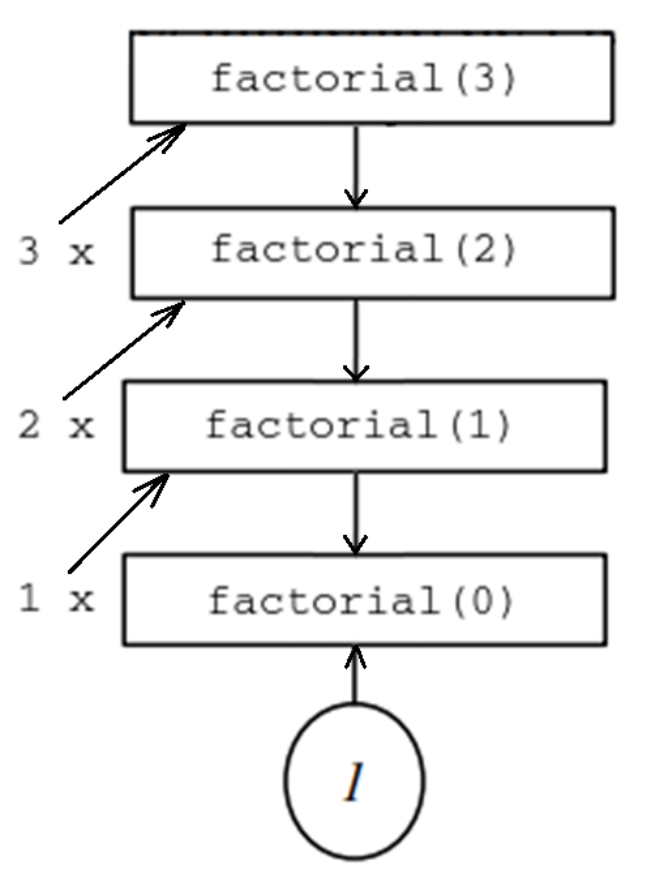
- Ejemplo 1: Calculo del factorial
- Solución en pseudocódigo

```
función Factorial (n)
inicio
    si n = 0
        entonces Factorial ←1
        sino Factorial ← n * Factorial (n-1)
    fin_si
fin
```

Ejemplos

- Ejemplo 1: Calculo del factorial de 3, descripción de llamadas apiladas:
  - factorial(3) -> 3 * factorial(2)
  - factorial(2) -> 2 * factorial(1)
  - factorial(1) -> 1 * factorial(0)
  - factorial(0) <- 1

- Ejemplo 1: Calculo del factorial de 3, descripción de llamadas apiladas:



Ejemplos

- Ejemplo 2: Leer una lista de números e imprimirla en orden inverso
  - Imprimir_al_reves(N) = Leer(N) e Imprimir_al_reves(N-1)
  - Imprimir_al_reves(N-1) = Leer(N-1) e Imprimir_al_reves(N-2)

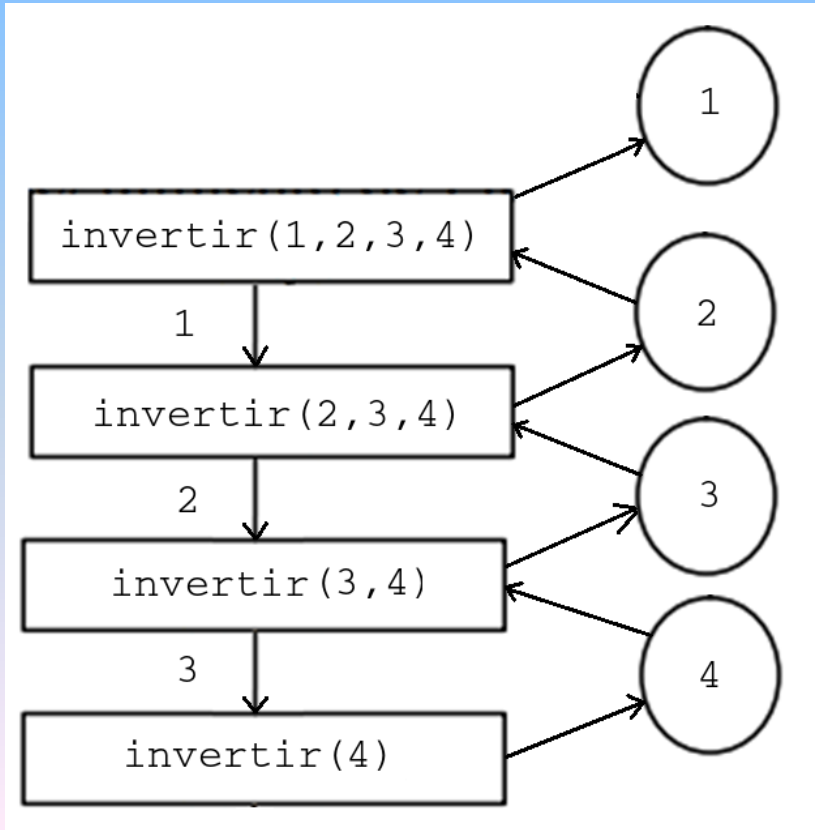
Ejemplos

- Ejemplo 2: Leer una lista de números e imprimirla en orden inverso
- Solución en pseudocódigo

```
algoritmo INVERTIR
inicio
  leer N
  si (N <> FF) entonces          { hay más números }
      INVERTIR
  escribir N
fin
```

Ejemplos

- Ejemplo 2: Imprimir la lista: 1, 2, 3, 4. Descripción de llamadas apiladas:



Ejemplos

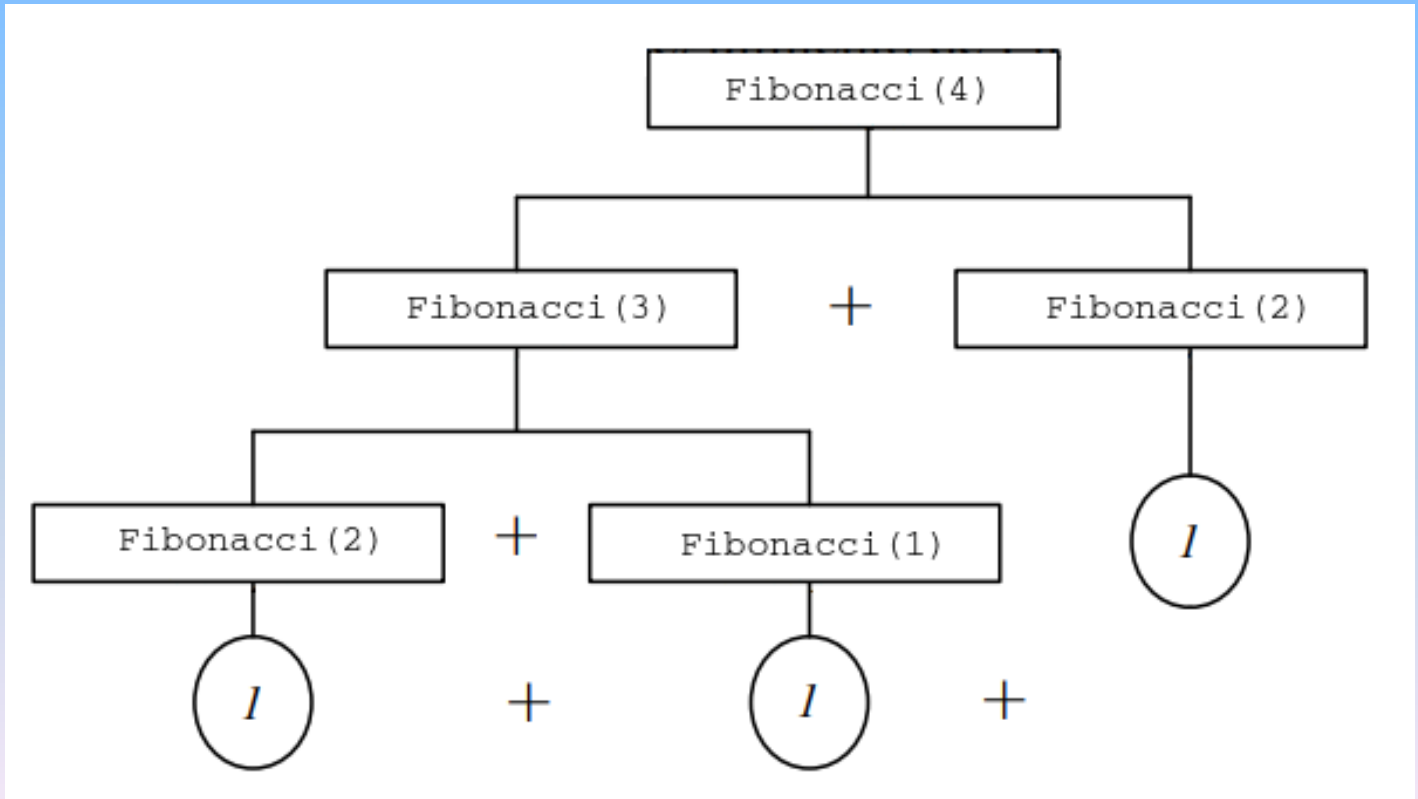
- Ejemplo 3: Serie de Fibonacci
  - Fibonacci(N) = Fibonacci(N - 2) + Fibonacci(N - 1)
    - Fibonacci(N - 2) = Fibonacci(N - 4) + Fibonacci(N - 3)
    - Fibonacci(N - 1) = Fibonacci(N - 3) + Fibonacci(N - 2)

Ejemplos

- Ejemplo 3: Serie de Fibonacci
- Solución en pseudocódigo

```
función FIBONACCI(n)
inicio
    si (n=1) o (n=2)
        entonces
            FIBONACCI ← 1
        sino
            FIBONACCI ← FIBONACCI (n-2) + FIBONACCI (n-1)
    fin_si
fin_función
```

- Ejemplo 1: Calculo del 4 numero de Fibonacci, descripción de llamadas apiladas:



Arreglos

- Estructura de datos contiguas en memoria
- Conjunto de datos del mismo tipo
- Componentes = Elementos
- Indices o subindices
  - nombreArreglo[2]

Arreglos

- Numero de elementos (tamaño) se especifica al declararlo.
- Numero de dimensiones
- Rango = Limites máximo y mínimo

Arreglos

- Tipos según numero de dimensiones
  - Unidimensionales (vectores)
  - Bidimensionales (matrices)
  - Multidimensionales

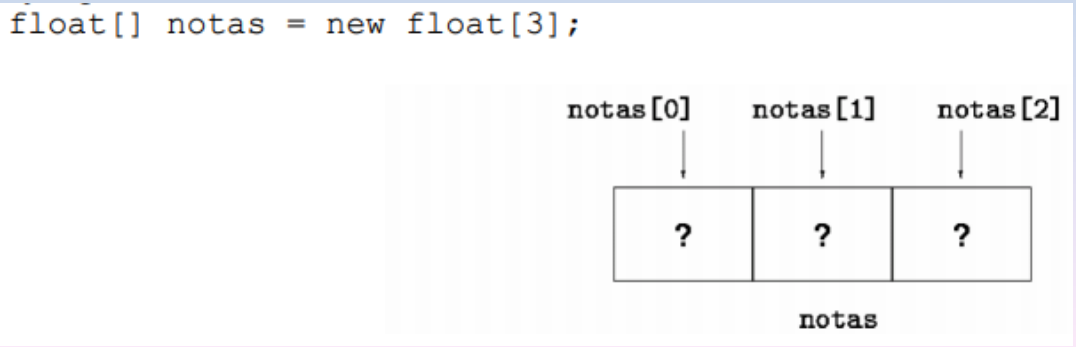
Arreglos

- Operaciones con arreglos
  - Declarar:
    - int[] vector = new int[5]; o int[] vector = {1,2,3,5,7} ;
    - int[][] matriz = new int[2][3]; o int[][] matriz = {{1,2,3},{4,5,6}};

Arreglos

- Operaciones con arreglos
  - Declarar:
    - int[] vector = new int[5]; o int[] vector = {1,2,3,5,7} ;
    - int[][] matriz = new int[2][3]; o int[][] matriz = {{1,2,3},{4,5,6}};
  - Acceder / Asignar:
    - vector[1] = 5;
    - matriz[2][1]=10;

```
float[] notas = new float[3];
```



Arreglos

- Operaciones con arreglos
  - Obtener tamaño:
    - vector.length
    - matriz.length -> Numero de filas
    - matriz[0].length -> Numero de columnas
  - Eliminar elemento
    - Mantener continuidad
  - Insertar elemento
    - Posición de inserción

Arreglos

- Cosas a tener en cuenta
  - Arreglos se pasan por parámetros por referencia
  - Al igualar arreglos no se crea uno nuevo

Cadenas de caracteres

- Almacenamos una palabra o frase
- Diferentes a arreglos de caracteres
  - char[] vc = new char[10];
  - char[] vc = {‘p’, ’a’, ‘l’, ‘a’, ‘b’, ‘r’, ‘a’};
- Strings en java
- No son modificables

Cadenas de caracteres

- Operaciones con Strings:
  - Declarar
    - String str = new String("contenido");
    - String str = "contenido" ;
  - Concatenar
    - String str = str1 + str2 ;
  - Longitud
    - str.length()

Cadenas de caracteres

- Operaciones con Strings:
  - Extracción
    - str.charAt(1);
  - Búsqueda
    - str.indexOf(char); str.indexOf(" cadena");
    - str.indexOf(char, posInicio);
    - Str.lastIndexOf(char); str.lastIndexOf(" cadena");

Cadenas de caracteres

- Tipo StringBuffer
  - Similar a String pero modificable
  - StringBuffer strb=new StringBuffer(“cadena”);
  - strb.SetcharAt(posición,nuevo carácter);
  - strb.insert(posicion,’carácter’);
  - strb.insert(posicion,”cadena”);