ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

ECYS | DIDT
DESARROLLO DE TRANSFERENCIA TECNOLÓGICA

Día, Fecha:	Jueves, 10/08/2023
Hora de inicio:	07:10

Introducción a la programación y computación 1 [F]

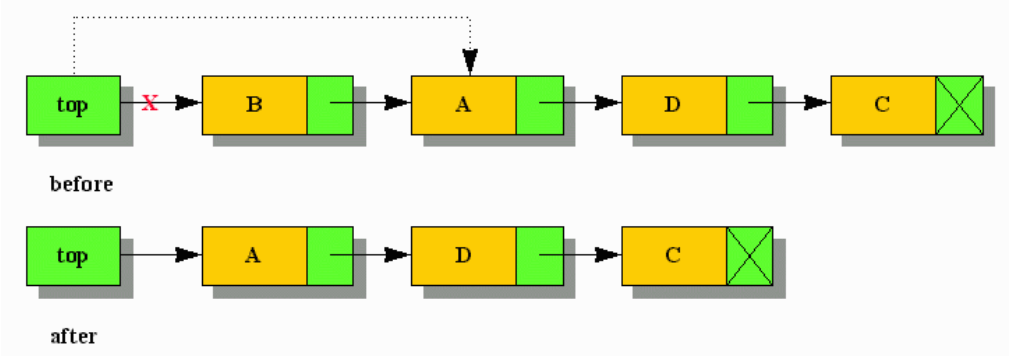
José Eduardo Morales García

Contenido

1. **Listas Dinámicas**
2. Métodos o Procedimientos
3. Funciones
4. Recursividad
5. Manejo de memoria
6. Métodos de Ordenamiento

Listas dinamicas

- Las listas dinámicas se implementan utilizando diferentes clases proporcionadas por la Biblioteca de Colecciones (java.util). Estas clases proporcionan métodos y funcionalidades para trabajar con listas que pueden cambiar de tamaño dinámicamente.



Listas dinamicas

- ArrayList: ArrayList es una implementación de la interfaz List que se basa en un arreglo dinámico que crece automáticamente según sea necesario.

```java
import java.util.ArrayList;

// Crear un ArrayList de tipo String
ArrayList<String> miLista = new ArrayList<>();

// Añadir elementos a la lista
miLista.add("Elemento 1");
miLista.add("Elemento 2");
miLista.add("Elemento 3");
```

Listas dinamicas

- LinkedList: Es una implementación de la interfaz List que se basa en una lista doblemente enlazada.

```java
import java.util.LinkedList;

// Crear una LinkedList de tipo Integer
LinkedList<Integer> miListaEnlazada = new LinkedList<>();

// Añadir elementos a la lista
miListaEnlazada.add(10);
miListaEnlazada.add(20);
miListaEnlazada.add(30);
```

Contenido

1. Listas Dinámicas
2. **Métodos o Procedimientos**
3. Funciones
4. Recursividad
5. Manejo de memoria
6. Métodos de Ordenamiento

Métodos o Procedimientos

Un **procedimiento**, o **método** es un conjunto de instrucciones como un mini programa que se ejecuta dentro de un programa, a solicitud.

- **Parámetros**, un parámetro en si es un objeto o valor que sirve como entrada o sirve para la ejecución y construcción de un valor como salida.
- **Bloque de instrucción,** es el conjunto de instrucciones contenidas dentro del mismo, la conforman diversos tipos de instrucciones entre ellas las estructuras de control

Métodos o Procedimientos

```java
// un metodo puede contener o no parametros
public void suma(int a, int b)
{
  /*

    Dentro de un metodo pueden venir diversos conjuntos
    de instrucciones o bloques.

  */
  int suma = a + b;
  System.out.println("El valor de la suma es: " + suma);

}
```

Contenido

1. Listas Dinámicas
2. Métodos o Procedimientos
3. **Funciones**
4. Recursividad
5. Manejo de memoria
6. Métodos de Ordenamiento

Funciones

Una **función** es un conjunto de instrucciones como un mini programa que se ejecuta dentro de un programa, a solicitud, las cuales a diferencia de los métodos nos retornaran un valor conforme al tipo de dato que representa.

- **Parámetros**, un parámetro en si es un objeto o valor que sirve como entrada o sirve para la ejecución y construcción de un valor como salida.
- **Bloque de instrucción,** es el conjunto de instrucciones contenidas dentro del mismo, la conforman diversos tipos de instrucciones entre ellas las estructuras de control

Funciones

```java
// una funcion puede contener o no parametros
public int suma(int a, int b)
{
  /*

    Dentro de una funcion pueden venir diversos conjuntos
    de instrucciones o bloques.

  */
  int suma = a + b;
  System.out.println("El valor de la suma es: " + suma);
  return suma;
}
```

Contenido

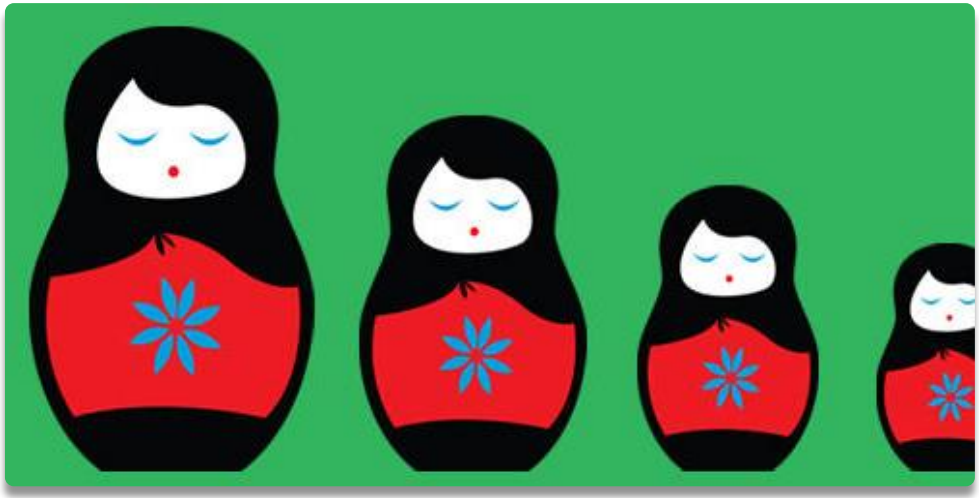
1. Listas Dinámicas
2. Métodos o Procedimientos
3. Funciones
4. **Recursividad**
5. Manejo de memoria
6. Métodos de Ordenamiento

Recursividad

- La recursividad es un concepto fundamental en matemáticas y en computación.
- Es una alternativa diferente para implementar estructuras de repetición (ciclos). Los módulos se hacen llamadas recursivas.

Recursividad

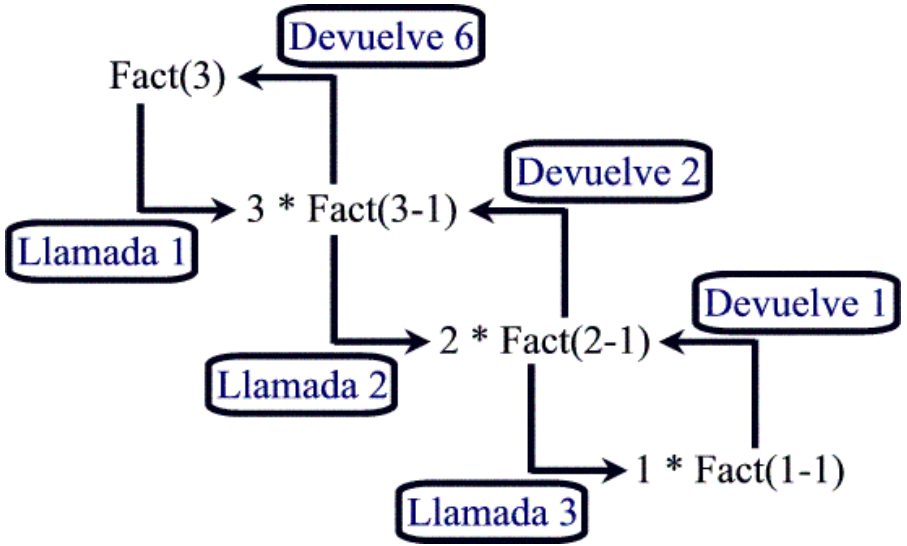
- La recursividad se compone de uno o varios casos base.
- Cada caso base es una solucion simple



Recursividad

- Caso recursivo: una solución que involucra volver a utilizar la función original, con parámetros que se acercan más al caso base.
- Los pasos que sigue el caso recursivo son los siguientes:
  - 1. El procedimiento se llama a sí mismo
  - 2. El problema se resuelve, tratando el mismo problema, pero de tamaño menor
  - 3. La manera en la cual el tamaño del problema disminuye asegura que el caso base eventualmente se alcanzará

Recursividad



Ventajas

- ✓ Solución elegante
- ✓ Abstracción

Desventaja

- ➢ Uso intensivo de memoria
- ➢ Potencial de bucles infinitos

Contenido

1. Listas Dinámicas
2. Métodos o Procedimientos
3. Funciones
4. Recursividad
5. **Manejo de memoria**
6. Métodos de Ordenamiento

Manejo de memoria

- "Stack" (pila) y "Heap" (montículo) son dos áreas de memoria importantes que se utilizan para gestionar la memoria en la ejecución de programas en muchos lenguajes de programación

Manejo de memoria

"Stack" (pila)

- La pila es una región de memoria organizada en forma de pila (LIFO: Last In, First Out).
- Se utiliza para almacenar variables locales, argumentos de función y otros datos relacionados con la ejecución de funciones o métodos.
- Cada vez que una función o método se llama, se crea un marco de pila (stack frame) que contiene información sobre las variables locales, los parámetros y otros datos relacionados con la ejecución de la función.
- Cuando la función finaliza su ejecución, su marco de pila se elimina y se libera automáticamente la memoria asociada a ese marco.
- La pila es de tamaño fijo y tiene un límite, lo que significa que la cantidad de memoria que se puede asignar en la pila es limitada.

Manejo de memoria

Heap (montículo):

- El montículo es una región de memoria utilizada para almacenar objetos y datos que necesitan ser asignados dinámicamente durante la ejecución del programa.
- A diferencia de la pila, el montículo es un área de memoria más grande y no tiene una estructura organizada en forma de pila o cola. En su lugar, utiliza un algoritmo de asignación más complejo para administrar la memoria y asegurarse de que los objetos se almacenen y se liberen de manera eficiente.
- Los objetos en el montículo pueden ser accedidos desde diferentes partes del programa y su tiempo de vida no está limitado por el alcance de una función o método.
- La gestión de memoria en el montículo debe realizarse explícitamente, es decir, el programador es responsable de asignar memoria para los objetos (mediante new) y también de liberarla cuando los objetos ya no son necesarios (mediante el recolector de basura de Java).

Contenido

1. Listas Dinámicas
2. Métodos o Procedimientos
3. Funciones
4. Recursividad
5. Manejo de memoria
6. **Métodos de Ordenamiento**

Métodos de ordenamiento para arreglos

- ORDENAMIENTO POR BURBUJA
- ORDENAMIENTO POR SELECCIÓN
- ORDENAMIENTO POR INSERCIÓN
- ORDENAMIENTO RÁPIDO (QUICK SORT)

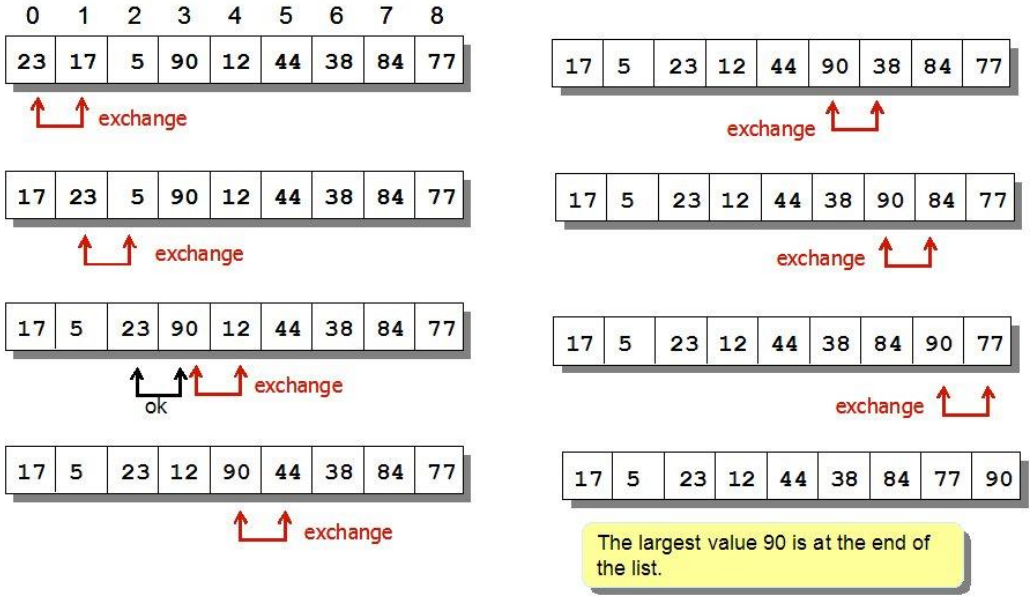
Ordenamiento por burbuja

- El ordenamiento por burbuja es un algoritmo simple que compara cada elemento del arreglo con su siguiente elemento, intercambiándolos si están en un orden incorrecto. Este proceso se repite varias veces hasta que el arreglo está completamente ordenado. Este método es fácil de implementar.

Algoritmo

1. Comenzamos con una lista de elementos desordenados que queremos ordenar.
2. Comparamos el primer elemento con el siguiente elemento en la lista. Si el primer elemento es mayor que el siguiente, se realiza un intercambio entre ellos.
3. Continuamos comparando y haciendo intercambios entre elementos adyacentes a lo largo de toda la lista, avanzando desde el inicio hasta el final de la lista. Esto provocará que el elemento más grande se mueva hacia el final de la lista.
4. Después de la primera iteración, el elemento más grande estará en su posición correcta al final de la lista.
5. Repetimos el proceso para las iteraciones restantes. En cada iteración, el elemento más grande de los elementos aún no ordenados se colocará en su posición correcta.

Ejemplo


0 1 2 3 4 5 6 7 8
23 17 5 90 12 44 38 84 77
exchange
17 23 5 90 12 44 38 84 77
exchange
17 5 23 90 12 44 38 84 77
ok
exchange
17 5 23 12 90 44 38 84 77
exchange
17 5 23 12 44 90 38 84 77
exchange
17 5 23 12 44 38 90 84 77
exchange
17 5 23 12 44 38 84 90 77
exchange
17 5 23 12 44 38 84 77 90
The largest value 90 is at the end of the list.

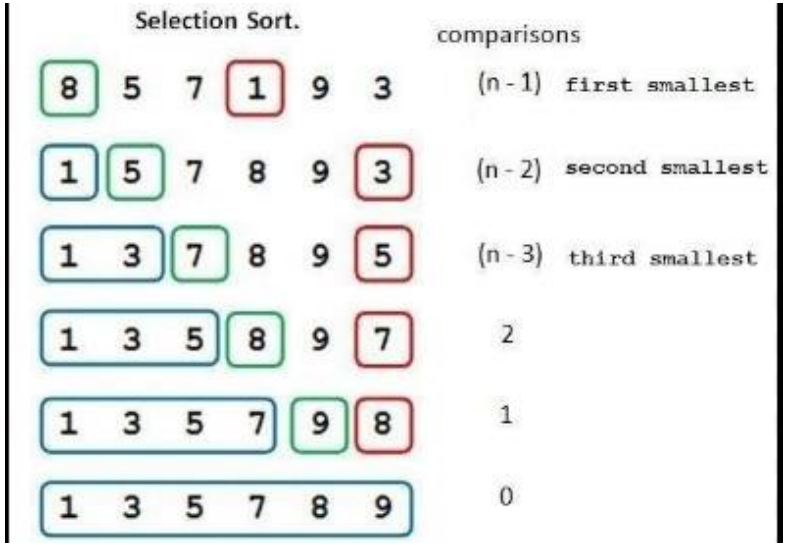
Ordenamiento por selección

- El ordenamiento por selección es un algoritmo que busca el elemento más pequeño del arreglo y lo intercambia con el primer elemento no ordenado. Luego, busca el segundo elemento más pequeño y lo intercambia con el segundo elemento no ordenado, y así sucesivamente. Este método es más eficiente que el ordenamiento por burbuja.

Algoritmo

- Comenzar con la lista de elementos desordenados.
- Encontrar el elemento más pequeño (o más grande) en la lista.
- Intercambiar el elemento más pequeño (o más grande) con el primer elemento de la lista no ordenada.
- Ahora, el primer elemento de la lista no ordenada se ha colocado en su posición correcta en la lista ordenada.
- Continuar el proceso con el resto de la lista no ordenada, que se ha reducido en un elemento debido a la colocación del elemento más pequeño (o más grande) en la lista ordenada.

Ejemplo


Selection Sort.
comparisons
8 5 7 1 9 3
(n - 1) first smallest
1 5 7 8 9 3
(n - 2) second smallest
1 3 7 8 9 5
(n - 3) third smallest
1 3 5 8 9 7
2
1 3 5 7 9 8
1
1 3 5 7 8 9
0

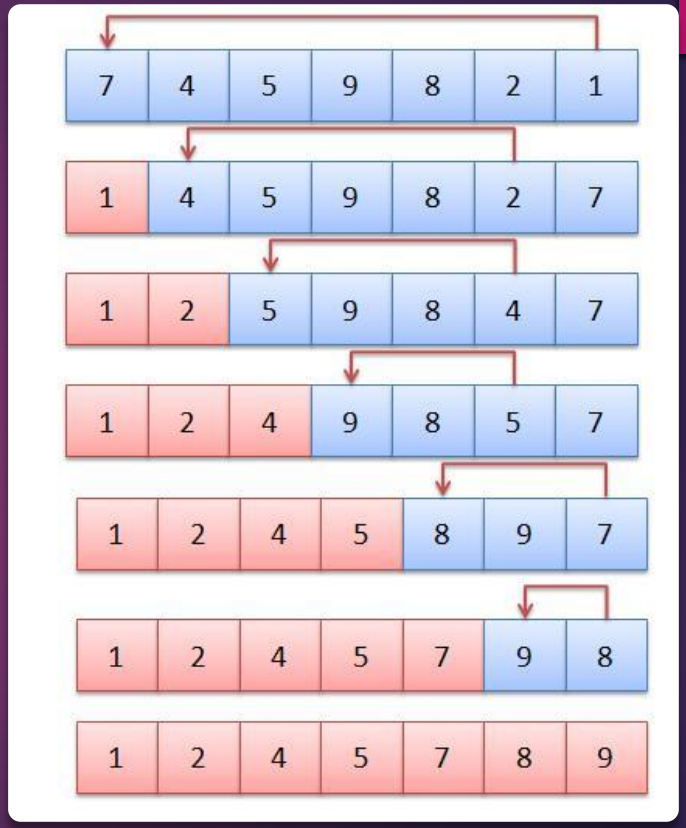
Ordenamiento por inserción

- El ordenamiento por inserción es un algoritmo que toma cada elemento no ordenado del arreglo y lo inserta en su posición correcta en el arreglo ordenado. Este método es similar al ordenamiento manual de un mazo de cartas y es más eficiente que el ordenamiento por burbuja y selección.

Algoritmo

- Comenzar con la lista de elementos desordenados.
- Dividir la lista en dos partes: una sublista ordenada y una sublista no ordenada. Al principio, la sublista ordenada contiene solo el primer elemento de la lista original y la sublista no ordenada contiene el resto de los elementos.
- Tomar el primer elemento de la sublista no ordenada y lo inserta en la posición correcta dentro de la sublista ordenada. Esto implica comparar el elemento con los elementos de la sublista ordenada y desplazar los elementos mayores que el elemento a la derecha para hacer espacio para insertar el elemento.
- Continuar el proceso con el siguiente elemento de la sublista no ordenada, insertándolo en su posición correcta dentro de la sublista ordenada.

Ejemplo



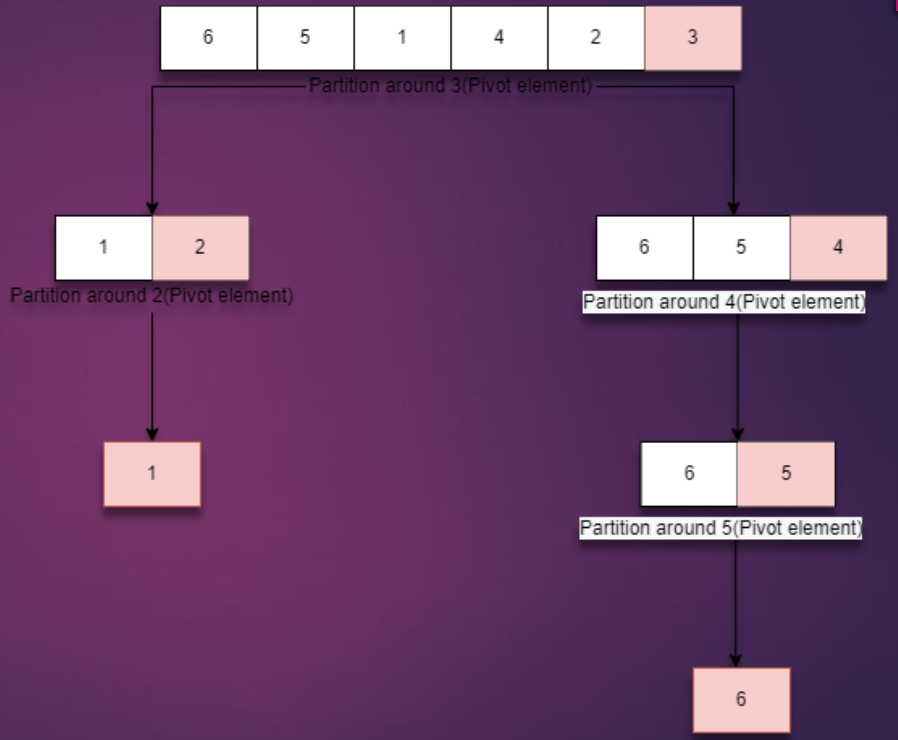
Ordenamiento rápido (Quick Sort)

- El ordenamiento rápido (Quick Sort) es un algoritmo de ordenamiento divide y vencerás que es muy eficiente en términos de complejidad. Este método selecciona un pivote del arreglo y divide el arreglo en dos partes, una con elementos menores al pivote y otra con elementos mayores al pivote. Luego, aplica el mismo proceso de forma recursiva a cada parte del arreglo hasta que todos los elementos estén ordenados.

Algoritmo

- Elegir un elemento de la lista como el pivote. El pivote puede ser cualquier elemento de la lista, pero se selecciona comúnmente el último elemento.
- Dividir la lista en dos subgrupos: un subgrupo de elementos más pequeños que el pivote y un subgrupo de elementos más grandes que el pivote.
- Colocar el pivote en su posición correcta en la lista, de manera que todos los elementos a la izquierda del pivote sean menores o iguales que el pivote, y todos los elementos a la derecha del pivote sean mayores que el pivote. Esto se conoce como la operación de particionamiento.
- Repetir los pasos 1 a 3 para cada subgrupo, de manera recursiva. Es decir, aplicar el mismo proceso de particionamiento en cada subgrupo hasta que todos los subgrupos tengan un solo elemento (que ya están ordenados).
- Combinar los subgrupos ordenados para formar la lista ordenada.

Ejemplo


6
5
1
4
2
3
Partition around 3(Pivot element)
1
2
Partition around 2(Pivot element)
6
5
4
Partition around 4(Pivot element)
1
6
5
Partition around 5(Pivot element)
6

Dudas y Preguntas

Parte practica