

Programación 2

Grado en Estadística Aplicada

Curso 2012-2013

Ordenación.

Jesús Correas – jcorreas@fdi.ucm.es

**Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid**

Ordenación

- La ordenación de datos es una parte importante de la programación, especialmente en aplicaciones estadísticas
 - ▶ Por ejemplo, es necesario para calcular percentiles o la mediana de una muestra de datos.
- Existen diversos algoritmos de ordenación, que dependen de diversos factores:
 - ▶ Si el volumen de datos es muy grande y no cabe en la memoria del ordenador se utiliza la **ordenación externa** (sobre disco o, antiguamente, en cintas magnéticas), con técnicas específicas.
 - ▶ Nosotros veremos la **ordenación interna**, sobre la propia memoria del ordenador. Los datos a ordenar están **almacenados en un array**.
- Vamos a ver tres técnicas para ordenar un array de números:
 - ▶ utilizando el **método de la burbuja**.
 - ▶ utilizando el **método de inserción**.
 - ▶ utilizando la técnica recursiva **mergesort**.
- Existen muchas otras técnicas, con distinta **eficiencia**.
- Los programas que vamos a ver se pueden modificar fácilmente para ordenar arrays con otros tipos de datos.

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

27	10	12	20	25	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	27	12	20	25	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	27	12	20	25	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	27	20	25	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	27	20	25	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	27	25	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	27	25	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	27	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	27	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	13	27	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	13	27	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	13	15	27	22
----	----	----	----	----	-----------	-----------	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	13	15	27	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	13	15	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	13	15	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	13	15	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	13	15	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	13	15	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	13	15	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	13	15	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	25	13	15	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	13	25	15	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	13	25	15	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	13	15	25	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	13	15	25	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	13	15	22	25	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	13	15	22	25	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	13	15	22	25	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	13	15	22	25	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	13	15	22	25	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	20	13	15	22	25	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	13	20	15	22	25	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	13	20	15	22	25	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	13	15	20	22	25	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	13	15	20	22	25	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	13	15	20	22	25	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de la burbuja

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array comparando cada elemento con el siguiente que está inmediatamente después.
 - ▶ Si el primer elemento es mayor al segundo, se intercambian.
 - ▶ A continuación se compara el segundo elemento con el que aparece a continuación y se realiza el mismo proceso.
 - ▶ Los elementos con mayor valor se van desplazando hacia la derecha como si fueran una **burbuja**.
 - ▶ Este proceso se debe repetir tantas veces como sea necesario, pues con un solo recorrido los elementos no quedan ordenados.

10	12	13	15	20	22	25	27
----	----	----	----	----	----	----	----

(Hay más iteraciones que no se muestran)

- ¿Cómo podemos hacer esto en C++?

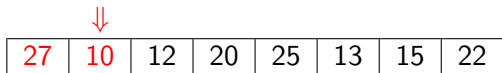
El método de la burbuja

```
const int N = 100;
typedef int tVector[N];

void burbuja(tVector v, int n) {
    int aux;
    for (int i = 1; i <= n; i++) {
        for (int j = 0; j < n-i; j++) {
            if (v[j] > v[j+1]) {
                aux = v[j];
                v[j] = v[j+1];
                v[j+1] = aux;
            }
        }
    }
}
```


El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.



- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	27	12	20	25	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	27	12	20	25	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓ ↓

10	12	27	20	25	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	12	27	20	25	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	12	20	27	25	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	12	20	27	25	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓ ↓

10	12	20	25	27	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	12	20	25	27	13	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	12	20	25	13	27	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	12	20	25	13	27	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	12	20	13	25	27	15	22
----	----	----	-----------	-----------	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	12	20	13	25	27	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	12	13	20	25	27	15	22
----	----	-----------	-----------	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	12	13	20	25	27	15	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓ ↓

10	12	13	20	25	15	27	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓ ↓

10	12	13	20	25	15	27	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	12	13	20	15	25	27	22
----	----	----	----	-----------	-----------	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	12	13	20	15	25	27	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

↓

10	12	13	15	20	25	27	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

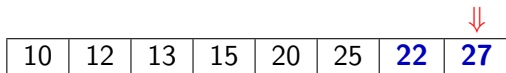
↓

10	12	13	15	20	25	27	22
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.



10	12	13	15	20	25	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

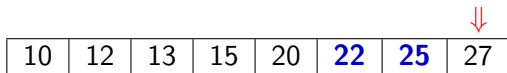
↓

10	12	13	15	20	25	22	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.



10	12	13	15	20	22	25	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

- Esta técnica consiste en lo siguiente:
 - ▶ Se recorre el array a ordenar desde la **segunda posición** hasta el final.
 - ▶ Por cada uno de los elementos (por ejemplo, $v[i]$), se recorre la parte del array que está a su izquierda y se coloca ese elemento ordenado respecto a los que están a la izquierda de i .
 - ★ Esto se hace comparando $v[i]$ con cada elemento que está a su izquierda e intercambiando su posición si $v[i]$ es menor.
 - ▶ Cuando se termina de hacer esto con todos los elementos del array, éste está ordenado.
- En cada iteración del bucle principal, los elementos que están a la izquierda del elemento a colocar **siempre están ordenados**.

10	12	13	15	20	22	25	27
----	----	----	----	----	----	----	----

- ¿Cómo podemos hacer esto en C++?

El método de inserción

```
const int N = 100;
typedef int tVector[N];

void insercion(tVector v, int n) {
    for (int i = 1; i < n; i++) {
        int x = v[i];
        int j = i-1;
        while ((j >= 0) && (x < v[j])) {
            v[j+1] = v[j];
            j--;
        }
        v[j+1] = x;
    }
}
```

Ordenación por mezcla: *mergesort*

- El subprograma anterior funciona correctamente, pero en el peor de los casos **su complejidad es cuadrática**, porque hay dos bucles anidados que pueden llegar a hacer del orden de N iteraciones.
- se pueden utilizar algoritmos más eficientes. Veremos el método de **ordenación por mezcla: mergesort**.
- Es un **método recursivo**. La idea es la siguiente:
 - ▶ En cada etapa nos centramos en **rango del array**. Inicialmente el rango es **todo el array**.
 - ▶ Para ordenarlo, primero lo dividimos en dos partes de aproximadamente el mismo tamaño.
 - ▶ Ordenamos cada una de esas dos partes. Para ello utilizamos **dos llamadas recursivas**.
 - ▶ Una vez ordenadas cada una de las mitades, **mezclamos** los datos de esas dos mitades.
- El método mergesort se basa en la idea de que mezclar dos arrays ordenados es **más sencillo** que ordenarlos.

Ordenación por mezcla: *mergesort*

- Dada la siguiente lista inicial:

27	10	12	20	25	13	15	22
----	----	----	----	----	----	----	----

Ordenación por mezcla: *mergesort*

- Dada la siguiente lista inicial:

27	10	12	20	25	13	15	22
----	----	----	----	----	----	----	----

- Divide:

27	10	12	20
----	----	----	----

25	13	15	22
----	----	----	----

Ordenación por mezcla: *mergesort*

- Dada la siguiente lista inicial:

27	10	12	20	25	13	15	22
----	----	----	----	----	----	----	----

- Divide:

27	10	12	20
----	----	----	----

25	13	15	22
----	----	----	----

- Divide:

27	10
----	----

12	20
----	----

25	13
----	----

15	22
----	----

Ordenación por mezcla: *mergesort*

- Dada la siguiente lista inicial:

27	10	12	20	25	13	15	22
----	----	----	----	----	----	----	----

- Divide:

27	10	12	20
----	----	----	----

25	13	15	22
----	----	----	----

- Divide:

27	10
----	----

12	20
----	----

25	13
----	----

15	22
----	----

- Divide:

27

10

12

20

25

13

15

22

Ordenación por mezcla: *mergesort*

- Dada la siguiente lista inicial:

27	10	12	20	25	13	15	22
----	----	----	----	----	----	----	----

- Divide:

27	10	12	20
----	----	----	----

25	13	15	22
----	----	----	----

- Divide:

27	10
----	----

12	20
----	----

25	13
----	----

15	22
----	----

- Divide:

27

10

12

20

25

13

15

22

- Combina los resultados:

10	27
----	----

12	20
----	----

13	25
----	----

15	22
----	----

Ordenación por mezcla: *mergesort*

- Dada la siguiente lista inicial:

27	10	12	20	25	13	15	22
----	----	----	----	----	----	----	----

- Divide:

27	10	12	20
----	----	----	----

25	13	15	22
----	----	----	----

- Divide:

27	10
----	----

12	20
----	----

25	13
----	----

15	22
----	----

- Divide:

27

10

12

20

25

13

15

22

- Combina los resultados:

10	27
----	----

12	20
----	----

13	25
----	----

15	22
----	----

- Combina los resultados:

10	12	20	27
----	----	----	----

13	15	22	25
----	----	----	----

Ordenación por mezcla: *mergesort*

- Dada la siguiente lista inicial:

27	10	12	20	25	13	15	22
----	----	----	----	----	----	----	----

- Divide:

27	10	12	20
----	----	----	----

25	13	15	22
----	----	----	----

- Divide:

27	10
----	----

12	20
----	----

25	13
----	----

15	22
----	----

- Divide:

27

10

12

20

25

13

15

22

- Combina los resultados:

10	27
----	----

12	20
----	----

13	25
----	----

15	22
----	----

- Combina los resultados:

10	12	20	27
----	----	----	----

13	15	22	25
----	----	----	----

- Combina los resultados:

10	12	13	15	20	22	25	27
----	----	----	----	----	----	----	----

Ordenación por mezcla: *Mergesort*

- El subprograma principal es el siguiente:

```
void mergesort(tVector v, int ini, int fin) {  
    if (ini < fin) {  
        int mitad = (ini + fin) / 2;  
        mergesort(v, ini, mitad);  
        mergesort(v, mitad+1, fin);  
        merge(v, ini, mitad, fin);  
    }  
}
```

- Los parámetros `ini` y `fin` delimitan el **rango del array** que se va a ordenar en cada paso.
- Inicialmente se debe llamar de la siguiente forma:

```
mergesort(vec, 0, N-1); // N-1 es la última posición de vec.
```

- `merge` mezcla los datos del vector que están ordenados desde `ini` hasta `mitad` y desde `mitad+1` hasta `fin`.
- ¿Cómo funciona `merge`?

10	12	20	27
----	----	----	----

13	15	22	25
----	----	----	----

10	12	13	15	20	22	25	27
----	----	----	----	----	----	----	----

Ordenación por mezcla: *Mergesort*

```
void merge(tVector v, int ini, int mitad, int fin) {
    int tamaño = fin - ini + 1;
    int w[tamaño]; // Se utiliza un array
    int i = ini; // auxiliar para mezclar.
    int j = mitad + 1;
    int k = 0;
    while ((i <= mitad) && (j <= fin)) { // Mezcla los datos de las
        if (v[i] < v[j]) w[k++] = v[i++]; // dos regiones del array.
        else w[k++] = v[j++];
    }

    if (i <= mitad) // Copia los datos que quedan
        for (; i <= mitad; i++) w[k++] = v[i]; // de la primera region
    else
        for (; j <= fin; j++) w[k++] = v[j]; // o de la segunda region.

    for (int i = ini; i <= fin; i++) // Copia los datos del array
        v[i] = w[i-ini]; // auxiliar al array original.
}
```

Ordenación por mezcla: *Mergesort*

- El código del método de ordenación por mezcla es más complicado del de ordenación por inserción.
- Sin embargo, se puede demostrar que, para un array de n elementos, **la eficiencia de mergesort está en el orden de $n \lg n$.**
- Esto es muy relevante para la ordenación de grandes cantidades de datos.
- Por ejemplo, con el programa de ejemplo se puede comprobar que si se ordena un vector de 100,000 números generados aleatoriamente, mergesort es aproximadamente **400 veces más rápido** que el método de inserción:

```
tiempo insercion: 21.78 segundos.  
tiempo mergesort: 0.05 segundos.
```