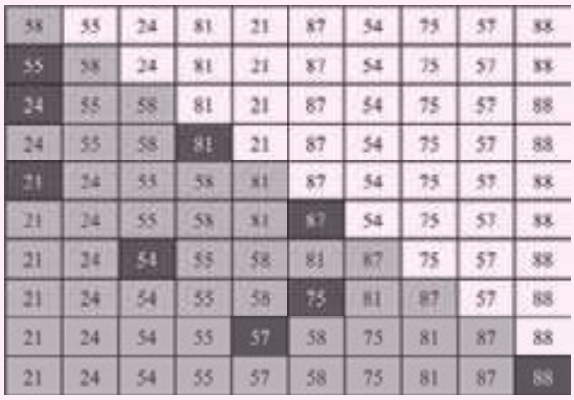
4.4.3. **Ordenamiento por Inserción**

Uno de los métodos más naturales para ordenar una secuencia de valores consiste en separar la secuencia en dos grupos: una parte con los valores ordenados (inicialmente con un solo elemento) y otra con los valores por ordenar (inicialmente todo el resto). Luego, vamos pasando uno a uno los valores a la parte ordenada, asegurándonos de que se vayan colocando ascendentemente, tal como se ilustra en el ejemplo 11.

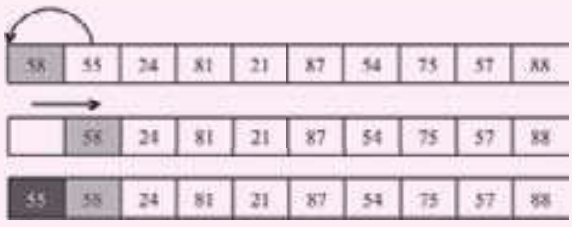
Ejemplo 11

Objetivo: Mostrar el funcionamiento del algoritmo de ordenamiento por inserción.

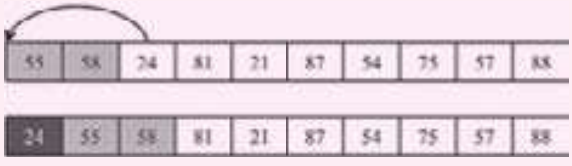
En este ejemplo se muestra cada una de las etapas por las que pasa el algoritmo de ordenamiento por inserción para ordenar ascendentemente un arreglo de valores de tipo simple.



- En este ejemplo, tenemos inicialmente un arreglo no ordenado de 10 posiciones (primera fila de la figura).
- Al inicio, podemos suponer que en la parte ordenada del arreglo hay un elemento (58). No está todavía en su posición final, pero siempre es cierto que una secuencia de un solo elemento está ordenada.
- Luego tomamos uno a uno los elementos de la parte no ordenada y los vamos insertando ascendentemente en la parte ordenada. Comenzamos con el 55 y vamos avanzando hasta insertar el valor 88.
- En la figura, en cada iteración aparece marcado el elemento que acaba de ser insertado.

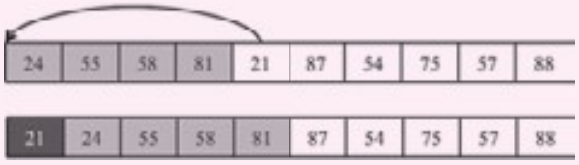


- Primera iteración: En la parte ordenada de la secuencia sólo está el valor 58. Vamos a insertar a esa parte el valor 55. Debemos buscar el punto en el que dicho valor debería encontrarse para que esa parte siga ordenada y desplazar los elementos necesarios que ya se encuentran allí. En este caso debemos mover el valor 58 a la derecha para abrir el espacio necesario para el 55.



- Segunda iteración: La parte ordenada ya tiene los valores 55 y 58. Vamos a insertar allí el valor 24. Repetimos el mismo proceso de desplazamiento y le abrimos espacio a este valor en el punto adecuado, para que esta parte siga ordenada.

Tercera iteración: Insertar el valor 81 a la parte ordenada es simple, puesto que no hay que desplazar ningún valor, ya que el 81 es mayor que todos los valores presentes en la parte ordenada.

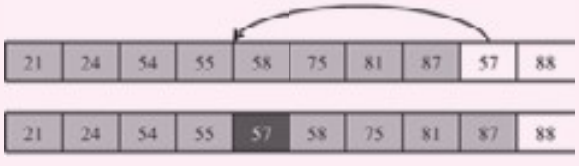


Cuarta iteración: Insertamos el valor 21, quedando en la primera posición del arreglo. Allí tenemos que desplazar a la derecha los cuatro valores que ya estaban ordenados.

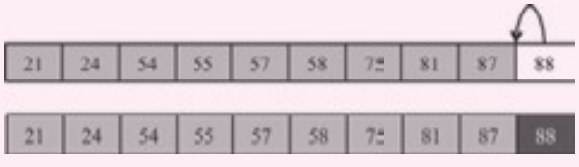
Quinta iteración: Insertamos el valor 87 sin necesidad de desplazar ningún elemento, puesto que debe quedar al final.

Sexta iteración: Para insertar el valor 54, debemos desplazar los valores que son mayores que él (55, 58, 81 y 87). Los demás (21, 24) no se mueven.

Séptima iteración: Vamos a insertar ahora el valor 75, para lo cual movemos hacia la derecha los valores 81 y 87.



Octava iteración: Para insertar el valor 57 movemos hacia la derecha los valores 58, 75, 81 y 87.



Novena iteración: La última iteración no implica ningún desplazamiento, puesto que el 88 es el mayor valor de todo el arreglo.

A continuación se muestra el código del método de la clase `Muestra` encargado de crear una instancia de la clase `MuestraOrdenada` usando la técnica de ordenamiento por inserción:

```
public MuestraOrdenada ordenarInsercion( )
{
    int[] arreglo = darCopiaValores( );

    for( int i = 1; i < tamanio; i++ )
    {
        for( int j = i; j > 0 && arreglo[ j - 1 ] >
arreglo[ j ]; j-- )
        {
            int temp = arreglo[ j ];
            arreglo[ j ] = arreglo[ j - 1 ];
            arreglo[ j - 1 ] = temp;
        }
    }

    return new MuestraOrdenada( arreglo );
}
```

- El ciclo externo tiene la responsabilidad de señalar con la variable "i" la posición del nuevo elemento que se va a insertar (hasta "i-1" está la parte ordenada).
- El ciclo interno va desplazando hacia abajo el elemento que se encontraba inicialmente en la posición "i", hasta que encuentra la posición adecuada.

Tarea 11

Objetivo: Utilizar la técnica de ordenamiento por inserción, para ordenar ascendentemente una secuencia de valores.

Suponiendo que los valores que se encuentran en la siguiente tabla corresponden a una secuencia de números enteros que quiere ordenar, muestre el avance en cada una de las iteraciones para el caso en el cual utilice la técnica de ordenamiento por inserción. Marque claramente en cada iteración la parte del arreglo que ya se encuentra ordenada.

25	13	45	12	1	76	42	90	56	27	33	69	72	99	81

Tarea 12

Objetivo: Medir el tiempo de ejecución de los algoritmos de ordenamiento e intentar identificar diferencias entre ellos.

Localice en el CD el ejemplo `n7_muestra`, cópielo al disco duro y ejecútelo. Siga luego las instrucciones que aparecen a continuación.

Algoritmo	5000	7000	10000	15000	20000	30000	40000	50000
Selección								
Burbuja								
Inserción								

Llene la tabla de tiempos de ejecución de los tres algoritmos de ordenamiento, para muestras de los tamaños allí especificados.

¿Qué se puede concluir de la tabla anterior?

Localice en el CD que acompaña al libro los entrenadores de ordenamiento allí disponibles, para complementar así lo estudiado en esta parte.

4.4.4. ¿Y Cuándo Ordenar?

Hay dos razones principales para ordenar información. La primera, para facilitar la interacción con el usuario. Si tenemos, por ejemplo, una lista de códigos de productos para que el usuario seleccione uno de ellos, es mucho más fácil para él encontrar lo que está buscando si le presentamos esta lista de manera ordenada. En ese caso no tenemos mayores opciones y debemos utilizar nuestros algoritmos de ordenamiento. La segunda razón es para hacer más eficientes los programas. Un método que busca información sobre una estructura que está ordenada gasta mucho menos tiempo que si los valores allí presentes no tienen ningún orden. El problema que se nos presenta aquí es que ordenar la información puede tomar un tiempo considerable; luego la pregunta que nos debemos hacer es: ¿cuándo es conveniente ordenar la información? La respuesta afortunadamente es simple y depende del número de búsquedas que se vayan a hacer. Si es una sola búsqueda, definitivamente no es un buen negocio ordenar antes la información, pero si vamos a hacer miles de ellas sobre el mismo conjunto de datos, la ganancia en tiempo amerita hacer un proceso previo de ordenamiento.

Más adelante veremos algunas estructuras de datos que están hechas para mantener permanentemente ordenada la información que contienen, lo que las hace ideales para manejar información sobre la cual hay búsquedas y modificaciones todo el tiempo.