PRÁCTICA Nº 3: 1 sesión

(del 27 de Marzo al 2 de Abril de 2003)

USO DEL TAD PILA

0. OBJETIVOS

El objetivo de esta práctica es la implementación del TAD Pila y su uso en la construcción de una versión (académica, no eficiente) del algoritmo de ordenación por inserción.

Utilización correcta en un programa de las dos implementaciones del TAD pila, de manera que, en su uso, sea transparente el tipo de implementación utilizado.

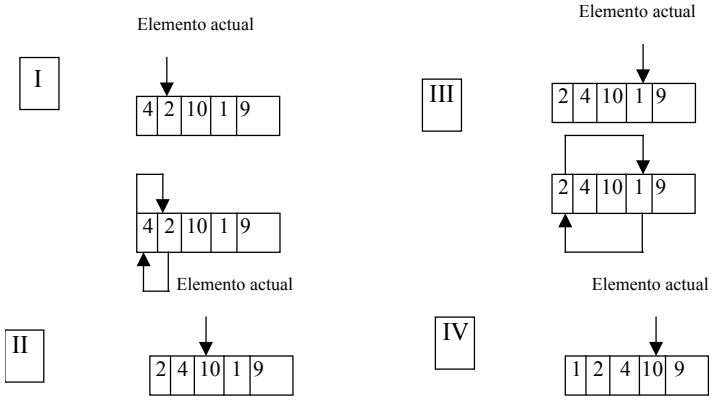
1. INTRODUCCIÓN

El algoritmo de ordenación por inserción es un algoritmo directo, de coste medio n^2 (donde n es el número de elementos a ordenar) basado en la comparación de elementos entre sí.

La idea es la siguiente: se trata de mantener un conjunto de elementos parcialmente ordenado que va creciendo de la siguiente manera:

- Dado un elemento k del conjunto a ordenar, se le busca el lugar que le corresponde según la relación de orden en el conjunto parcialmente ordenado.
- Se le asigna el correspondiente lugar en el conjunto parcialmente ordenado.
- Se repiten los dos pasos anteriores hasta que se termine el conjunto a ordenar.

El algoritmo eficiente en espacio, usa como estructura de datos el propio vector de elementos desordenados. Visualmente el algoritmo podría funcionar de la siguiente manera:



En esta práctica utilizaremos la TAD pila para conseguir el mismo funcionamiento. La idea es la siguiente:

Usaremos dos pilas `A` y `B` para obtener en cada momento, el lugar que le corresponde a cada nuevo elemento dentro del conjunto parcialmente ordenado. La pila `A` contendrá el conjunto parcialmente ordenado. La pila `B` servirá para almacenar temporalmente los elementos de la pila `A` que se desapilan de `A` para dejar hueco al nuevo elemento. Sin entrar en detalles, el algoritmo consistiría en los siguientes puntos. Dado un vector `V` de elementos desordenados:

1. Apilo en `A` el primer elemento de `V`
2. Paso al siguiente elemento de `V` mientras éste exista.
3. Desapilo de `A` todos los elementos mayores (o menores, según si queremos hacer la ordenación de mayor a menor o de menor a mayor) que el considerado y los voy apilando en `B`.
4. Apilo en `A` el elemento.

5. Desapilo de `B` todos los elementos y los apilo nuevamente en `A`.
6. Vuelvo al paso 2 hasta que recorra todo el vector.

Al acabar, en la pila `A` se encuentra el vector ordenado, siendo la cima de la pila el elemento mayor o menor (según el criterio de ordenación).

Por último se actualiza el vector `V` con el contenido de la pila `A`.

2. REALIZACIÓN DE LA PRÁCTICA

a) Implementación de la clase y comprobación de su correcto funcionamiento

En esta tercera práctica, se debe implementar una clase en C++ que permita representar el TAD pila, asumiendo la interfaz estándar vista en teoría:

```
class Pila
{
   public:
      Pila (void);                 // Constructor por defecto
      bool Apilar (Valor);
      bool Desapilar (void);
      bool CimaPila (Valor &);
      bool PilaVacia (void);
   private:
      .??.
};
```

Se construirán las dos implementaciones concretas vistas en teoría utilizando como nombres de archivos

`##PilaE.h` y **`##PilaE.cpp`** para la implementación estática.

`##PilaD.h` y **`##PilaD.cpp`** para la implementación dinámica.

Además se creará un programa completo llamado **`##insercion.cpp`** donde se construya el algoritmo de ordenación expuesto. Este algoritmo será un subprograma que será llamado por **`main()`**. El programa principal se encargará de iniciar aleatoriamente un vector de enteros de **`TALLAMAX`** y escribir en pantalla dicho vector antes de la ordenación y tras ésta. Este programa completo deberá funcionar tanto con la versión estática como con la versión dinámica de las pilas.

b) **Opcional:** Una mejora del algoritmo consiste en no vaciar completamente la pila B en cada nueva inserción de un elemento de V, sino ir trasvasando elementos de A a B o viceversa hasta lograr que en la pila A queden los elementos menores que el elemento a insertar y en la pila B queden los mayores.

Tras la última inserción pasaremos todos los elemetos de B a A, de manera que tendremos todos los elementos ordenados en A.

4. ENTREGA DE PROGRAMAS

Al comienzo de la siguiente sesión de prácticas se entregarán al profesor cinco o seis ficheros:

1) Ficheros de la clase **`Pila`** (**`##Pila.cpp`** y **`##Pila.h`**)
2) Fichero principal donde se realiza la ordenación (**`##insercion.cpp`**).
3)Opcional: programa opcional (**`##op_insercion.cpp`**).

Nota Muy Importante

Antes de poder empezar a realizar cualquiera de las prácticas es necesario presentar las hojas de especificación de programas (documentación de programas).

ENTREGA DE PROGRAMAS: Al comenzar la sesión de prácticas del 3 al 9 de Abril.