**Instituto Superior de Formación Técnica Nº 151 logo151-trans.gif  
Carrera: Analista de Sistemas  
1 Año. Algoritmos y Estructuras de Datos I**

|  |  |
| --- | --- |
| **Trabajo Práctico Nº 7** | **Unidad 7** |
| **Modalidad:** Semi-Presencial | **Estratégica Didáctica:** Trabajo Grupal. |
| **Metodología de Desarrollo:** acordar | **Metodología de Corrección:** acordar docente |
| **Carácter de Trabajo:** Obligatorio – con Nota | **Fecha Entrega:** A confirmar por el Docente. |

**Marco Teórico:**

1. Describir el concepto de Algoritmo de ordenación por **inserción**
2. ¿Qué diferencia tiene con el Algoritmo de ordenación por inserción con intercambios?
3. Dar pautas de Claves de ordenación
4. ¿Qué entiende por Estabilidad de la ordenación?
5. Describir la Ordenación por **selección** directa
6. ¿Cómo funciona el Método de la **burbuja**?
7. ¿Cómo gestionaría Listas Listas ordenadas y estructuras?
8. ¿Cómo se realizan las búsquedas en listas ordenadas?
9. ¿Qué es y cómo se Implementa la Búsqueda binaria?

**Marco Práctico:**

**Desarrollar un Programa** que:

1. La Directora nos Encarga llevar un registro de los Alumnos por carrera y año.  
   Para ello deberemos armar un Programa que guarde los datos de los Alumnos en Alumnos.txt.   
   Adicionalmente deberá realizar las siguientes Tareas:  
   1. Mostrar Listado de los Alumnos x pantalla
   2. Mostrar un Alumno Determinado (buscar y mostrar por pantalla)
   3. Insertar un Alumno
   4. Eliminar un Alumno
   5. Buscar un alumno
   6. Que permita Ordenar de Forma Ascendente y Descendente los Alumnos y Mostrarlos por pantalla
   7. Opcional - Que permita elegir el campo de Ordenamiento.

Algunas Consideraciones:

1. Usar Archivo para persistir y recuperar
2. Usar Estructuras, contador y arrays (max 100 alumnos)
3. Usar sobrecarga de Operadores  
   bool operator>(tRegistro opIzq, tRegistro opDer);   
   bool operator<(tRegistro opIzq, tRegistro opDer);
4. Modularizar en funciones la implementación
5. Mantener el Main lo más pequeño posible.

Lic. Oemig José Luis.

***Comentario***:

Hola Profe. El programa quedó un poco largo porque quise componer y reutilizar las estructuras creadas en el TP5. No implementé la sobrecarga de los operadores > y < pero sí está la opción de ordenar cada lista individualmente (por curso) en orden ascendente y descendente tanto por nombre como por apellido. El principal motivo por el que no lo usé es porque me daba la impresión de que los operadores < y > necesitaban conocer la estructura de mi lista. (¿no estaría violando eso el principio de ocultamiento de información?)

Adjunto el archivo Alumnos.txt con varios datos como para que pueda probar las distintas funcionalidades. Saludos.

1. Describir el concepto de Algoritmo de ordenación por **inserción.**

El Algoritmo de Ordenación por Inserción lo que hace es, dada una lista con valores, comienza a recorrerla. Y, a medida que va recorriendo la lista, va comparando el valor actual con todos sus antecesores. Si encuentra algún elemento mayor que el actual, lo que hace el algoritmo es un corrimiento de los elementos de la lista (de atrás hacia adelante) desde el primer elemento mayor encontrado hasta la posición actual e **inserta** el elemento actual en la posición del primer elemento mayor a éste encontrada.

1. ¿Qué diferencia tiene con el Algoritmo de ordenación por inserción con intercambios?

La diferencia es que en el ordenamiento por inserción, a medida que voy recorriendo, el elemento ‘actual‘ *se fija* y se va comparando este elemento uno a uno con todos sus antecesores y, cuando encuentra uno que es mayor que él se hace un corrimiento en toda la lista desde ese lugar hasta el actual y se inserta el elemento actual.

En cambio, en el Ordenamiento por inserción con intercambio lo que se hace es ir recorriendo la lista y, al posicionarse en el lugar ‘ *i* ’ lo que hace es comenzar desde la primera posición hasta la posición ‘ *i* ’ a comparar cada elemento con el que le sigue y, si no están en orden, los intercambia.

1. Dar pautas de Claves de ordenación

* Una de las claves para la ordenación es seleccionar un campo que permita la comparación.
* Si bien lo que debe compararse es un campo concreto de una lista de estructuras, el intercambio deberá realizarse entre estructuras completas.

1. ¿Qué entiende por Estabilidad de la ordenación?

Un ordenamiento estable mantiene el orden relativo que tenían originalmente los elementos con claves iguales. Por ejemplo, si una lista ordenada por fecha se reordena en orden alfabético con un algoritmo estable, todos los elementos cuya clave alfabética sea la misma quedarán en orden de fecha.

1. Describir la Ordenación por **selección** directa

Lo que hace el Algoritmo de Ordenación por selección directa es:

* Comenzar a recorrer la lista deteniéndose primero en la primera ubicación.
* Recorrer toda la lista buscando el elemento mínimo.
* Una vez hallado el mínimo, se intercambia con el primero.
* Luego se posiciona en el segundo lugar de la lista y busca el mínimo elemento desde el segundo hasta el último.
* Cuando lo encuentra, lo intercambia por el segundo.
* Se posiciona en el tercer lugar y repite el procedimiento.

1. ¿Cómo funciona el Método de la **burbuja**?

La técnica utilizada se denomina ordenación por burbuja debido a que los valores más pequeños “burbujean” gradualmente (suben) hacia la cima o parte superior del array de modo similar a como suben las burbujas en el agua, mientras que los valores mayores se hunden en la parte inferior del array.

1. ¿Cómo gestionaría Listas Listas ordenadas y estructuras?

En caso de realizar búsquedas, optaría por la búsqueda binaria. Y en caso de tener que reordenar, utilizaría alguno de los algoritmos vistos (la versión para estructuras)

1. ¿Cómo se realizan las búsquedas en listas ordenadas?

Recorremos hasta encontrarlo o mayor / al final mediante (probablemente) algún método rápido de búsqueda en listas ordenadas como búsqueda binaria.

1. ¿Qué es y cómo se Implementa la Búsqueda binaria?

Se sitúa el índice de búsqueda en el centro de la lista y se comprueba si nuestra clave coincide con el valor del elemento central. Si no se encuentra el valor de la clave, se sitúa en la mitad inferior o superior del elemento central de la lista. Así se continúa subdividiendo en dos cada sublista hasta dar con el elemento deseado o cuando coinciden el límite inferior con el superior.