Soluciones propuestas

Pregunta teórico-práctica

Métodos de búsqueda. Enumere y explique brevemente los distintos métodos de búsqueda que conozca.
Apartado 9.3 del libro de texto.

Diseñe un módulo que permita buscar un elemento dentro de un vector ordenado con datos de tipo cadena. El método deberá buscar sólo entre las posiciones a y b.

```
entero: función Buscar(E vector:v;E cadena:el;E entero:a,b)
var
   entero: izq, der, cen
inicio
   izq ← a
   der ← b
   repetir
      cen ← (izq + der) div 2
      si v[cen] > el entonces
         der ← cen - 1
      si_no
         si v[cen] < el entonces
            izq ← cen + 1
         fin_si
      fin_si
   hasta_que (v[cen] = el) o (izq > der)
   si v[cen] = el entonces
       devolver(cen)
   si_no
       devolver(0)
   fin_si
fin_función
```

Problema

En una tabla de NxN elementos están almacenadas las distancias en kilómetros entre distintas ciudades. Los nombres de las ciudades se almacenan en un array de N elementos de tipo cadena.

Array con las distancias

Ciudades	1	2	3	4	5
1		621	352	395	563
2	621		349	694	809
3	352	349		633	620
4	395	694	633		958
5	563	809	620	958	

Array con las ciudades

1	Madrid
2	Barcelona
3	Valencia
4	Bilbao
5	Almería

Se desea:

- Definir las estructuras de datos necesarias para realizar la aplicación.
- Diseñar una función que, a partir de dos nombres de ciudades que se pasan como argumento, devuelva la distancia entre ellas.
- Diseñar un procedimiento que obtenga una lista con las ciudades y las distancias, ordenada de la más cercana a la más lejana a una ciudad dada que introduciremos como argumento. Por ejemplo, si la ciudad que se pasa es Almería, la lista sería:

Madrid	563
Valencia	620
Barcelona	809
Bilbao	958

Definición de las estructuras de datos

```
const
   N = 5
tipos
   //Tabla para las distancias
   array[1..N, 1..N] de entero = tabla
  //Vector para las ciudades
  array[0..5] de cadena = vector
  //Array de registros para el apartado 3 de la práctica
  registro = ciudad
     cadena : nombre
     entero : distancia
  fin_registro
  array[0..N-1] de ciudad = ciudades
```

Distancia entre dos ciudades

```
entero : función DistanciaEntreCiudades(E tabla : t; E vector : v ;E entero : n;
                                        E cadena : ciud1,ciud2)
// t es la tabla de distancias
// v es el vector con el nombre de las ciudades
// n es el número de elementos
// ciud1 y ciud2 son las ciudades
var
   entero : pos1, pos2 //Posición de la ciudad1 y ciudad2 en el vector de ciudades
inicio
   pos1 ← buscar(v, ciud1,n)
   pos2 ← buscar(v, ciud2,n)
   devolver(t[pos1,pos2])
fin_función

entero: función Buscar(E vector:v;E cadena:el;E entero:n)
//Método de búsqueda secuencial con centinela
var
   entero: i
inicio
   i ← n
   v[0] ← el
   mientras el <> v[i] hacer
      i ← i - 1
   fin_mientras
fin_función
```

Lista de ciudades

```
procedimiento ListaCiudades(E tabla : t; E vector : v; E entero : n; E cadena : ciud;
                            E/S ciudades : lista)
// t es la tabla de distancias
// v es el vector con el nombre de las ciudades
// n es el número de elementos
// ciud es la ciudad que se pasa como argumento
// lista es el array donde van a aparecer ordenadas las ciudades
var
   entero : i,j,pos
inicio
   //Rellenar ciudades y distancias en el array de registros
   //En el de registros array no debe aparecer la ciudad que se pasa como argumento
   //Por eso se busca su posición en el array
   pos ← buscar(v,ciud,n)
   desde i ← 1 hasta n-1 hacer
      //Si i=pos, se trata de la ciudad que no se debe incluir
      si i <> pos entonces
          lista[i].distancia ← t[i,pos]
          lista[i].nombre ← v[i]
      fin_si
```

```
   fin_desde

   Ordenar(lista,n-1)
fin_procedimiento

procedimiento Ordenar(E/S ciudades:v;E entero : n)
//Ordenación por el método de inserción directa
//El vector v tiene elementos entre 0 y n
var
   entero : i,j
inicio
   desde i ← 2 hasta n hacer
      v[0] ← v[i]
      j ← i - 1
      mientras v[i].distancia > v[0].distancia hacer
          v[j+1] ← v[j]
          j ← j - 1
      fin_mientras
      v[j] ← v[0]
fin_desde
fin_procedimiento
```