



ALGORITMOS Y ESTRUCTURAS DE DATOS I

- Septiembre 2002 -

Dpto. Ing. Electrónica, Sistemas Informáticos y Automática
E.P.S. La Rábida (Universidad de Huelva)

Apellidos :	DNI :
Nombre :	Grupo :

NOTAS

- Todas las **preguntas** deberán contestarse en **folios distintos**, escritos a bolígrafo, con letra clara y sin tachones. No se corregirá ninguna respuesta que no cumpla estos requisitos. Se dispone de 20 minutos antes de que comience a contar la convocatoria.
- El examen tendrá una duración de **2h y 30 minutos**.
- Se establece un tiempo límite para preguntar dudas con respecto a los enunciados de los ejercicios de una hora.
- **Todas las preguntas deben estar perfectamente comentadas, indicando el razonamiento seguido hasta llegar a la implementación dada.**
- Se deberá dejar el D.N.I. o cualquier otro documento identificativo válido (T. U. O, Carnet de conducir) visible al profesor durante el tiempo que sea necesario.
- El examen deberá ser firmado por el alumno.

Pregunta 1. (4 Puntos)

Partiendo de la siguiente declaración de tipos:

tipo

```
pnode = puntero a nodo;  
nodo = registro  
        dato: entero;  
        sig : pnode;  
fregistro  
lista = pnode; (* Esta lista no será circular *)
```

ftipo

y contando con la existencia de los siguientes subprogramas ya implementados:

funcion ANTERIOR (L:Lista; pos:pnode):pnode;

{ Devuelve la dirección del elemento anterior al apuntado por el parámetro **pos**, dentro de la lista **L**. }

accion BORRAR (var L:lista; pos:pnode);

{ Borra de la lista **L**, el elemento apuntado por el parámetro **pos** }

realizar un subprograma (acción o función) para cada una de las siguientes actividades:

- a) Invertir los elementos de una lista. (**1,5 puntos**)
- b) Obtener la posición del elemento mayor de una lista. Si el valor mayor se encuentra repetido dentro de la lista, el alumno debe tomar la determinación de elegir uno de ellos, comentando dicha decisión en la implementación que realice. (**1 punto**)
- c) Utilizando el subprograma del apartado b), ordenar de menor a mayor los elementos de una lista. (**1,5 puntos**)

Todas las funciones deben estar perfectamente comentadas, indicando:

- el razonamiento seguido hasta llegar a la implementación dada,
- quienes son cada uno de los parámetros y variables locales declaradas,
- comentario de aquellas líneas de código que necesiten explicación o no queden lo suficientemente claras.

Pregunta 2. (3 Puntos)

Se desea diseñar una acción o función recursiva que permita obtener el Número máximo de una tabla de números enteros positivos, así como el N° de veces que aparece dicho número dentro de la tabla.

Para ejemplificar el use de la acción o función diseñada anteriormente, construya un algoritmo que haga una llamada a la función o acción y utilice los valores devueltos para mostrarlos por pantalla.

La función o acción a desarrollar se denominara **Vecesmaximo**.

Pregunta 3. (3 Puntos)

Para guardar la información referente a los empleados que hay en una empresa, se utilizara un **fichero de acceso directo** con el tipo de registro siguiente (el primer registro del fichero es el numero 0):

```
empleado = registro
           dni: entero;
           nomapell: tabla[1..50] de caracter;
           dirección: tabla[1..30] de caracter;
           sueldo: real;
           libre: carácter; (* L -> libre; O -> ocupado *)
           fregistro
```

La **organización que tiene dicho fichero** es la siguiente:

Dada la información de un empleado la posición que le corresponde en el fichero es el numero de DNI. Ya que el rango de valores distintos que puede dar el DNI es tan grande, se deberá implementar un método de direccionamiento que asocie a cada DNI un valor comprendido en el rango de 0 a 199.

Con el fin de evitar las colisiones (ya que a varios DNI les puede corresponder el mismo lugar), se mantendrá una zona de desbordamientos dentro del mismo fichero, en concreto desde el registro 200 hasta el registro 250 sin ningún orden.

Desarrolle las siguientes acciones y funciones

a) accion inicializar (f : fichero de empleado); (0,5 Puntos)

(* Este procedimiento pondrá en todos los registros del fichero f el campo *libre* a L, indicando que todos los registros están libres para ser ocupados cuando se inserte un registro *)

b) accion altas (f: fichero de empleado; r: empleado); (2,5 Puntos)

(* Si el registro r se encuentra previamente en el fichero dará un mensaje de error. Si no está, lo insertara en el fichero en el lugar que le corresponda de acuerdo con la organización descrita anteriormente, poniendo el campo *libre* del registro con el valor O, que indica ocupado *).



ALGORITMOS Y ESTRUCTURAS DE DATOS I

SOLUCIÓN DEL EXAMEN

- Septiembre 2002 -

*Dpto. Ing. Electrónica, Sistemas Informáticos y Automática
E.P.S. La Rábida (Universidad de Huelva)*

PROBLEMA 1

a) Invertir los elementos de una lista

accion INVERTIR (var L:Lista);

(* Obtendremos el primer elemento de la lista **L**, lo eliminaremos de dicha lista y posteriormente lo insertaremos como primer elemento de la lista auxiliar **laux**. Este proceso se repetirá mientras existan elementos en la lista inicial **L**. Como la lista **L** es quien debe quedar invertida, al final de este proceso se igualaran **L** y **laux**. *)

var

laux: lista; (* Lista auxiliar en la que iremos elaborando la lista L invertida *)
primero: lista; (* Puntero al primer elemento de la lista L en cada momento *)
nuevo: pnodo; (* Nuevo nodo, que sera insertado al principio, de la lista laux que contiene el valor del campo dato del primer elemento de la lista L *)

fvar

inicio

laux:=nulo;

mientras (L <> nulo) hacer

inicio

primero := L;

reservar (nuevo);

nuevo ^.dato := primero^.dato;

BORRAR (L, primero);

(* Insertamos nuevo como el primer elemento de laux *)

nuevo^.sig := laux;

laux := nuevo;

fin;

L := laux; (* En este momento L queda invertida *)

faccion

b) Obtener la posición del elemento mayor de una lista

funcion MAYOR (L: lista): pnode;

(* Recorreremos la lista **L** con el puntero **actual** e iremos comparando el dato apuntado por dicho puntero con el dato más grande leído hasta el momento, que estará apuntado por **grande**. Si el dato apuntado por **actual** es mayor, el puntero **grande** apuntará donde apunte actual. *)

var

grande, actual: pnode;

fvar

inicio

grande := L; actual := L;

mientras (actual < > nulo) hacer

inicio

si (actual^.dato > grande ^.dato) → grande := actual;

 □ (actual^.dato <= grande ^.dato) → nada;

 actual := actual^.sig;

fin

retorna (grande);

ffuncion

c) Utilizando el subprograma del apartado b), ordenar de menor a mayor los elementos de una lista

accion ORDENAR (var L: lista);

(* Obtendremos el dato más grande de la lista **L** en cada momento utilizando la función **MAYOR**, y lo insertaremos como primer elemento de una lista auxiliar **laux** en la que iremos construyendo la lista ordenada de menor a mayor (si el elemento más grande de **L** lo insertamos al principio de **laux**, al finalizar el proceso **laux** está ordenada de menor a mayor). Hecho esto, eliminamos dicho elemento de la lista **L**. El proceso finaliza cuando **L** esté vacía. Como último paso haremos que **L** apunte donde apunta **laux**, momento en el cual **L** estará ordenada de menor a mayor. *)

var

laux: lista;

masgrande: pnode; (* Contiene la dirección del elemento mayor de la lista L en cada momento *)

nuevo: pnode; (* Su campo dato contiene el valor mayor de la lista L en cada momento. Este nodo será insertado como primer elemento de la lista laux *)

fvar

inicio

laux := nulo;

mientras (L < > nulo) hacer

inicio

 masgrande := MAYOR (L);

 reservar (nuevo);

 nuevo ^.dato := masgrande ^.dato;

 BORRAR (L, masgrande);

 (* Insertamos nuevo como primer elemento de laux *)

 nuevo ^.sig := laux;

 laux := nuevo;

fin;

L := laux; (* En este momento la lista L queda ordenada *)

faccion

PROBLEMA 2

Como nos indican diseñar una función o acción que a partir de una tabla debe devolver dos valores (máximo y veces que aparece éste), esto nos determina que debemos usar una acción.

Una de las posibles implementaciones que puede tener esta acción es la que se muestra a continuación, donde se sigue la idea de obtener el máximo de la tabla (en las sucesivas llamadas recursivas) y posteriormente, una vez obtenido el máximo, ir contando cuantas veces aparece este. Para ello se establece que siempre se analiza la posición **min**, haciendo que la posición en análisis se incremente en uno de una llamada a otra.

accion vecesmaximo(tab:t; min,max:entero; var maximo,veces:entero);

(* **T** es la tabla de enteros que se pasa como parámetro a la acción. Puede entenderse la misma como $T=tabla[1..20]$ de entero, por ejemplo, **min** y **max** son los límites de la tabla que en este momento estamos analizando, **maximo** es la variable donde se almacenara el maximo o número mayor de la tabla, **veces** es la variable donde se almacenara el N° de veces que esta presente el maximo en la tabla *)

inicio

si $min > max \rightarrow veces := 0;$

(* si ya no hay tabla que tratar, ponemos veces a 0. Este constituye el caso base que corta la recursividad *)

□ $min \leq max \rightarrow$

inicio

si (tab[**min**] > maximo) \rightarrow

maximo:=tab[**min**];

(* Si el elemento que esta en la posición **min** es mayor que el **maximo**, el es el nuevo **maximo**. En esta comparación se debe partir de un **maximo** ya existente. La primera vez que se haga use de la accion debemos velar que la variable **maximo** venga convenientemente inicializada a 0 (el menor de los números enteros positivos) *)

□ (tab[**min**] <= maximo) \rightarrow nada;

vecesmaximo(tab,min+1,max,maximo,veces);

(* tratado el elemento de la posición **min**, pasamos a analizar el elemento de la posición min+1 *)

si (tab[**min**] = maximo) \rightarrow

veces:=veces+1

(* Si el elemento de la posición en analisis (**min**) es igual al **maximo** que se ha obtenido, se acumula 1 en el n° de veces que aparece *)

□ (tab[**min**] <> maximo) \rightarrow nada;

fin

faccion

algoritmo pregunta2;

const

MIN= 1;

MAX=20; (* por ejemplo *)

fconst

tipo

t=tabla[MIN..MAX] de entero;

ftipo

var

ta:t;

i:entero;(* variable auxiliar para la carga inicial de la tabla a analizar *)

maximo,veces:entero; (* para almacenar el maximo y el n° de veces de este *)

fvar

inicio

para i:= MIN hasta MAX hacer

inicio

escribir("Derne un valor");

leer(ta[i]);

fin

maximo:=0; (* Por efecto inicializamos **maximo** al menor de los enteros positivos *)

vecesmaximo (ta,MIN,MAX,maximo,veces);

escribir(" El maximo es ",maximo," y aparece ",veces," veces");

falgoritmo

PROBLEMA 3

accion inicializar (f : fichero de empleado);

var

r : empleado; i: entero

fvar

inicio

r.libre:= 'L';

para i := 0 hasta 250 hacer escribir(f, i, r);

faccion:

accion altas (f: fichero de templeado; r: templeado);

var

i: entero;
rl: templeado;
encontrado, encontradohueco : booleano;

fvar

inicio

i := r.dni mod 200; (* este será su sitio principal entre 0 y 199*)
leer (f, i, rl); (* leo para ver quien esta en su sitio*)

si rl.libre = 'L' →

inicio (* esta vacío, entonces lo escribo allí*)

r.libre:= 'O';
escribir(f, i, r);

fin

□ rl.libre='O' → (* su sitio está ocupado*)

si rl.dni = r.dni → escribir('ERROR, el registro ya esta en el fichero'); (* esta ocupado, pero por el mismo*)

□ rl.dni <> r.dni

inicio

(* no es él, puede estar en la zona de desbordamientos, lo busco allí *)

i := 200; encontrado := falso;

mientras (i < 251) y (no encontrado) hacer

inicio

leer (f, i, rl);

si rl.libre = 'L' → i := i + 1

□ rl.libre = 'O' →

si rl.dni = r.dni → encontrado := cierto

□ rl.dni <> r.dni → i := i + 1;

fin;

si encontrado → escribir(' ERROR, ya esta en el fichero ')

□ no encontrado →

(* no esta tampoco en la zona de desbordamientos del fichero, busco el primer hueco para insertarlo allí*)

inicio

encontradohueco := falso; i := 200;

mientras (i < 251) y (no encontradohueco) hacer

inicio

leer (f, i, r l);

si rl.libre ='L' → encontradohueco := cierto

□ rl.libre ='O' → i := i + 1;

fin;

si encontradohueco →

inicio (* lo pongo aqui*)

r.libre := 'O';

escribir(f, i, r);

fin

□ no encontradohueco → escribir('ERROR, fichero lleno');

(* no hay ningun hueco en el zona de desbordamientos *)

fin;

fin;

facción;