



# ALGORITMOS Y ESTRUCTURAS DE DATOS I

- Junio 2002 -

## PARTE I

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

Apellidos :

DNI :

Nombre :

Grupo :

### Sección Informativa:

- La duración de la parte I es de 45 minutos.
- Se dispone de 15 minutos antes de que comience a contar la convocatoria.
- El tiempo de resolución de dudas es de 20 minutos a partir de que comience el examen.
- Puntuación : Ejercicio 1 : 2 ; Ejercicio 2 : 1

### 1.- Conteste V (Verdadero) o F (Falso) a las siguientes cuestiones.

Puntuación: Acierto: 0.1 Error: -0.1 Sin contestar: 0

- 1.- Un fichero es una estructura de datos homogénea, que puede almacenar tipos de datos estáticos o dinámicos.
- 2.- Es necesario declarar el fichero físico en nuestro algoritmo cuando queremos almacenar información en él.
- 3.- Si se abre un fichero en modo lectura /escritura se podrá leer de él, escribir en él, e incluso añadir datos al final.
- 4.- Sobre un fichero de acceso directo se puede hacer una lectura secuencial sin necesidad de especificar la posición en la instrucción *leer*.
- 5.- En la declaración de un fichero se especifica si se trata de un fichero de acceso directo o secuencial.
- 6.- Un fichero con sólo declararlo existe y podemos almacenar información en él.
- 7.- La escritura en una determinada posición, de un fichero de acceso directo desplaza los elementos que haya a continuación de él hasta el final del fichero.
- 8.- En un fichero con organización directa el orden físico en que fueron escritos sus elementos no tiene por qué corresponder con el orden en que se lean.

- 9.- El área de desbordamiento se utiliza cuando se producen colisiones (llaves sinónimas) en un fichero con organización secuencial indexada.
- 10.- Las estructuras dinámicas de datos están formadas por variables dinámicas.
- 11.- Las variables dinámicas, al igual que las estáticas, tienen un tipo y un tamaño que ha de ser fijado en la declaración de variables.

12.- Un fichero como una estructura de datos dinámica que es, puede crearse y destruirse en tiempo de ejecución por medio de una variable puntero.

13.- La siguiente declaración de tipos es correcta:  
tipo

```
tipo_dato = ..... ;  
tipo_elemento = puntero a tipo_dato;  
fich = fichero de tipo_elemento;
```

ftipo

- 14.- Cualquier identificador seguido del símbolo ^ será una variable dinámica referenciada por dicho identificador, que será una variable de tipo puntero.
- 15.- La asignación  $p^:= q^+ 3 * q^x$  es válida con una declaración de tipos y variables adecuada.
- 16.- Las variables dinámicas que forman una estructura de datos dinámica se alojan siempre en posiciones de memoria contiguas o adyacentes.
- 17.- La liberación de la variable dinámica referenciada por una variable puntero asigna a dicho puntero el valor nulo.
- 18.- Las variables de tipo puntero cuando son parámetros de una función se pasan siempre por valor o por referencia.
- 19.- La búsqueda en una estructura dinámica de datos que esté ordenada puede mejorarse mediante la aplicación del método de búsqueda binaria o dicotómica.
- 20.- En una lista doblemente enlazada podemos saber que está vacía evaluando la verdad o falsedad de la igualdad: inicio = fin, siendo inicio y fin los punteros externos de la lista.

## 2.- Cuestiones ( 1 punto)

Puntuación: Acierto: 0.1 Error: -0.1 sin contestar. 0

Dadas las siguientes declaraciones de tipos y variables:

tipo

```
nodo = registro
      dato:entero;
      sig:punt;
      fregistro
```

```
punt = puntero a nodo;
```

ftipo

var

```
p,q:punt;
```

fvar

Conteste F (Falso) si no son correctos o no tienen sentido los siguientes segmentos de código en un algoritmo y V (Verdadero) en caso contrario:

A.- reservar (p);  
p:= nulo;  
q:= p;

F.- reservar (q);  
p:= q;  
reservar (p);

B.- liberar(q);  
q^:= p^;

G.- q:=nulo;  
escribir (q^.dato);

C.- leer (p);  
q:= p;

H.- leer (p^.dato);  
p:= p^.sig;

D.- q^:= nulo;

I.- retorna (p^.sig = q^);

E.- liberar(p);  
p:=nulo;

J.- p^.sig^.sig:= q^.sig;



# ALGORITMOS Y ESTRUCTURAS DE DATOS I

- Junio 2002 -

## PARTE II

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

Apellidos :	DNI :
Nombre :	Grupo :

### Sección Informativa:

- **La duración de la parte 2 es de 1,15 minutos.**
- Se dispone de 15 minutos antes de que comience a contar la convocatoria.
- El tiempo de resolución de dudas es de 20 minutos a partir de que comience el examen.
- A la entrega, el alumno debe mostrar su DNI.
- Para superar el examen de teoría debe alcanzar una nota igual o mayor a 5.
- **Puntuación** : Ejercicio 3 : 3,5 ; Ejercicio 4 : 3,5

### 3.-

a) Escribir la función `Maxlista(list: lista; imin, imax: entero): entero`; la cual calcula de forma recursiva el elemento mayor de una lista comprendida entre `imin` e `imax` (índices mínimo y máximo). Para la realización de este ejercicio no podrán utilizarse ninguna otra acción ni función. El tipo `lista` tiene definido por:

tipo

Lista: tabla[ 1..100] de entero;

ftipo

b) Escribir un programa o algoritmo que cargue la lista desde el fichero secuencial `DATOS.DAT` de enteros, realice la llamada a `Maxlista` y muestre el resultado por pantalla. Suponemos que el fichero existe y contiene datos.

**4.- Dadas las siguientes primitivas de listas, donde `lis` siempre apunta al principio de la lista y `x` es un elemento de tipo carácter :**

*InsertarL(lis,x)*: inserta `x` al final de la lista apuntada por `lis`.

*BorrarL(lis,x)*: elimina `x` de la lista apuntada por `lis`.

*InicializaL(lis)*: inicializa la lista apuntada por `lis`.

*VaciaL(lis)*: devolverá verdadero o falso en función de si la lista apuntada por `lis` está vacía o no.

*PrimeroL(lis)*: devuelve una referencia al primer elemento de la lista apuntada por `lis`.

*UltimoL(lis)*: devuelve una referencia al último elemento de la lista apuntada por `lis`.

*ConsultarL(lis,x)*: devuelve en `x` la información que se almacena en el nodo apuntado por `lis`.

### Implemente:

- Las operaciones que definen el comportamiento de las pilas.
- Las operaciones que definen el comportamiento de las colas.
- Las operaciones `VaciarP` y `VaciarC` que eliminan el contenido de una pila y una cola respectivamente.

NOTA1: Sólo pueden utilizar las operaciones de listas indicadas anteriormente para implementar las operaciones de pilas y colas, puesto que desconoce la naturaleza de la implementación de las listas.

NOTA2: El tipo `lista` se encuentra implementado en un módulo externo y lo único que conoce es que `lis` es del tipo `lista`.



Universidad  
de Huelva

# ALGORITMOS Y ESTRUCTURAS DE DATOS I

## SOLUCIÓN DEL EXAMEN

- Junio 2002 -

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

### EJERCICIO 1

1	F	11	V
2	F	12	F
3	V	13	F
4	F	14	V
5	F	15	F
6	F	16	F
7	F	17	F
8	V	18	F
9	F	19	F
10	V	20	F

### EJERCICIO 2

A	F
B	F
C	F
D	F
E	V
F	F
G	F
H	V
I	F
J	V

### EJERCICIO 3

a) Existen diferentes implementaciones de la función recursiva MaxLista. De entre todas se ha elegido la expuesta a continuación por ser la que tiene más valor explicativo. Sin embargo, otras soluciones son igualmente válidas.

funcion MaxLista(list: lista; imin, imax: entero): entero;

inicio

si imin = imax → retorna (list[imin]); (\* si la lista tiene solo un elemento, éste será el mayor \*)

si list[imin] > MaxLista(list, imin+1,imax) → retorna (list[imin])  
(\* si la lista tiene más de un elemento, se elige el mayor entre el primero y el resto de la lista \*)

□ list[imin] <= MaxLista(list, imin+1,imax) → retorna(maxlista(list,imin+1,imax);

ffuncion

b)

algoritmo examen;

tipo

lista= tabla [1.. 100] de entero;

ftipo

var

T: lista;

f: fichero de enteros; (\* el fichero es de enteros, tal como indica el enunciado\*)

i,dato: entero;

fvar

inicio

i:=1; (\* inicializar I a 1\*)

abrir(f,1,'DATOS.DAT'); (\* abrir el fichero en modo lectura \*)

leer(f,dato);

(\* la tabla tiene una dimensión de 1 a 100, por tanto, habrá que contar cuantos elementos se van escribiendo en la tabla, para no sobrepasar esta dimensión \*)

mientras no fdf(f) y (i<=100) hacer

inicio

leer (f, dato);

T[i]:= dato; (\* escribir el dato leído del fichero en la tabla\*)

i:= i+1; (\* incrementar el índice\*)

fin;

si i > 100 → Escribir ('NO se ha cargado el fichero en su totalidad, por contener más de 100 entradas')

□ i <=100 → Escribir(maxlista(T,1,i-1));

cerrar(f); (\* cerrar el fichero\*)

falgoritmo.

## EJERCICIO 4

**a)**

accion InicializaP (var p:lista);  
inicio

InicializaL(p);

faccion:

funcion VacíaP (p:lista):booleano;  
inicio

retorna(VaciaL(p));

ffuncion:

accion tope (p:lista; var c:carácter);  
inicio

si no VacíaP(p) →  
    ConsultarL(UltimoL(p),c)  
□ VacíaP(p) → nada;

faccion:

**b)**

accion InicializaC (var p:lista);  
inicio

InicializaL(p);

faccion:

funcion VacíaC (cola:lista): booleano;  
inicio

retorna(VaciaL(cola));

ffuncion:

accion ConsultarC(cola:lista; var c:carácter);  
inicio

si no VacíaC(cola) →  
    ConsultarL(PrimeroL(cola),c)  
□ VacíaC(cola) → nada;

faccion:

**c)**

accion SacarP(var p:lista);  
var c:carácter; fvar  
inicio

mientras no VacíaP(p) hacer  
    POP(p,c)

faccion

accion PUSH (var p:lista;c:carácter);  
inicio

InsertarL(p,c);

faccion:

accion POP (p:lista; var c:carácter);  
inicio

si no VacíaP(p) →  
    inicio  
        Tope(p,c);  
        BorrarL(p,c);  
    fin  
□ VacíaP(p) → nada;

faccion:

accion MeterC (var cola: lista; c:carácter);  
inicio

Insertar(cola,c);

faccion:

accion SacarC (var cola:lista; var c:carácter);  
inicio

si no VacíaC(cola) →  
    inicio  
        ConsultarC(cola,c);  
        BorrarL(cola,c);  
    fin  
□ VacíaC(cola) → nada;

faccion:

accion SacarC(var cola:lista);  
var c:carácter; fvar  
inicio

mientras no VacíaC(cola) hacer  
    SacarC(cola,c);

faccion