



# LABORATORIO DE PROGRAMACIÓN II

Departamento de Lenguajes  
y Ciencias de la Computación  
UNIVERSIDAD DE MÁLAGA

E.T.S.I.Informática. 1º. 29-06-01

Duración del Examen: 4 horas

APELLIDOS \_\_\_\_\_ NOMBRE \_\_\_\_\_

GRUPO(A/B/C/D) \_\_\_\_\_ ESPECIALIDAD(Gestión/Sistemas) \_\_\_\_\_ ORDENADOR \_\_\_\_\_

La empresa de software *ModulaSoft, S.A.* se dedica a la elaboración y venta de componentes software. Cada componente software lleva la siguiente información: nombre del componente (32 caracteres) y lista de algoritmos, donde cada algoritmo, tendrá un nombre (32 caracteres) y el lenguaje en que ha sido implementado (32 caracteres). Para ello nos solicita el desarrollo de una aplicación de gestión llamada **CMPSOFT.MOD** (y que deberá estar **OBLIGATORIAMENTE** en el directorio C:\CMPSOFT) que tenga las siguientes opciones de Menú:

Elaborado por: <Nombre> <Apellidos>  
<Curso> <Grupo> <Especialidad>  
<Fecha> <Ordenador>

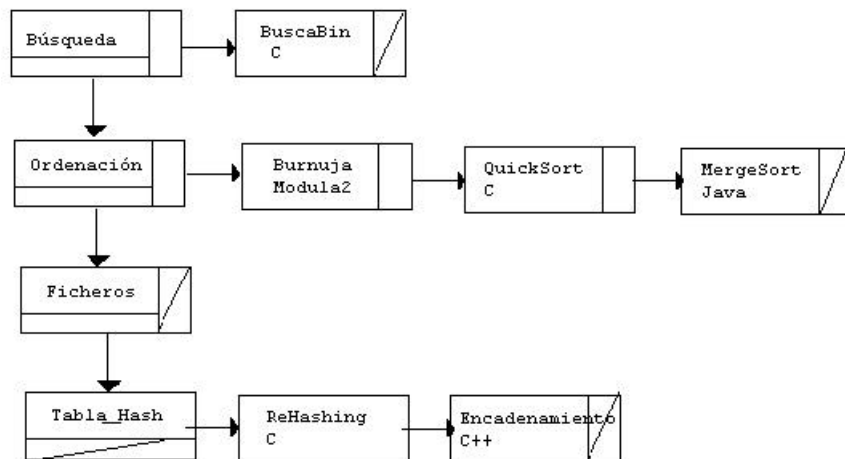
M E N U P R I N C I P A L  
=====

- A.- Introducir un Componente Software nuevo desde Teclado.
- B.- Mostrar la Información de un Componente Software.
- C.- Mostrar la Información de todos los Componentes Software.
- D.- Borrar un Componente Software.
- E.- Salvar los Datos en un fichero de Texto.
- F.- Cargar Datos desde un Fichero de Texto.
- G.- Mostrar la Información de todos los Algoritmos (nombre y componente) implementados en un determinado lenguaje.
- X.- Salir del Programa.

Introduzca Opción: \_

## Implementación:

Debido a que el número de componentes software y de algoritmos es a priori desconocido, usaremos una estructura (repositorio), como la de la figura, cuya **DEFINICIÓN** se adjunta y que **NO PODRÁ SER MODIFICADA**.



y debido al uso constante de cadenas de caracteres tendremos un módulo Mcadena (transparente) en el cual encapsularemos las operaciones relativas a las cadenas de caracteres, cuya definición se adjunta y **NO PODRÁ SER MODIFICADA.**

Opciones:

**A. Introducir un componente nuevo desde Teclado.**

Se pedirá al usuario el nombre del paquete. Si el componente ya existiera se informará de ello y no se insertará, si no existe con anterioridad, se solicita el número de algoritmos a introducir y los datos (nombre y lenguaje) de cada uno de ellos para insertarlos en la lista de algoritmos (en la que pueden haber duplicados).

**B. Mostrar la información de un Componente Software.**

Se pedirá al usuario el nombre del componente y se mostrará por pantalla todos los algoritmos que lo componen y en qué lenguaje está implementado cada uno de ellos. Si no existiera dicho componente, se informará de su no existencia.

**C. Mostrar por pantalla la información de todos los Componentes Software.**

Se mostrará por pantalla todos componentes almacenados y para cada uno de ellos, los algoritmos que lo componen y en qué lenguaje está implementado.

**D. Borrar un Componente.**

Se pedirá al usuario el nombre de un componente y se eliminará de la estructura junto con todos los algoritmos que contenga. Si no existiera se hará nada, ni se informará.

**E. Salvar los datos en un fichero de Texto.**

Se pedirá al usuario el nombre de un fichero y se grabará todo el contenido de la estructura en el formato que se adjunta. Si existiera previamente un fichero con dicho nombre se borrará su contenido y si no existiera se creará.

El formato del fichero será: 1 línea por componente con la siguiente información:

<componente> <numAlg> < alg 1> <leng1> < alg 2> <leng2> .... < algN> <lengN>

donde :

<componente> nombre del componente  
<numAlg> número de algoritmos que contiene  
<algi> nombre del algoritmo i-esimo  
<lengi> lenguaje del algoritmo i-esimo

Ejemplo: La figura anterior se salvaría:

```
Búsqueda 1 BuscaBin C
Ordenación 3 Burbuja Modula2 QuickSort C MergeSort Java
Ficheros 0
Tabla_Hash 2 Reshasing C Encadenamiento C++
```

**F. Cargar Datos desde un Fichero de Texto.**

Se pedirá al usuario el nombre de un fichero que tiene el mismo formato descrito para su escritura y se insertará su contenido en la estructura (que puede o no estar vacía). Si el fichero no existiera o estuviese vacío no se inserta nada ni se dará ningún error.

**G. Mostrar la Información de todos los Algoritmos.**

Se pedirá el nombre de un lenguaje al usuario y se mostrará por pantalla todos los algoritmos (nombre y componente) implementados en un dicho lenguaje.

**X. Salir del Programa.**

Pedirá confirmación de salida, liberará todos los recursos que hayan sido reservados y saldrá al sistema.

## Definiciones de Módulos que **NO PODRÁN SER MODIFICADAS**:

```
DEFINITION MODULE MCadena;  
FROM FIO IMPORT File;
```

```
CONST
```

```
    MAXCAD=32;  
    FINCAD=CHR(0);
```

```
TYPE
```

```
    TCadena    = ARRAY [1..MAXCAD] OF CHAR;  
    TSeparadores = SET OF CHAR;
```

```
PROCEDURE IgualCadena(cad1,cad2:TCadena):BOOLEAN;
```

```
(* Nos dice si las dos cadenas son iguales sin tener en cuenta  
    mayúsculas o minúsculas.
```

```
*)
```

```
PROCEDURE LeePalabra(fich:File; sep:TSeparadores):TCadena;
```

```
(* Lee una palabra de un fichero de texto cuyo descriptor  
    se le pasa como par metro. La palabra termina con un EOL  
    o un separador
```

```
*)
```

```
END MCadena.
```

```
DEFINITION MODULE MRepo;
```

```
FROM MCadena IMPORT TCadena;
```

```
TYPE
```

```
    TRepositorio;
```

```
PROCEDURE CrearRepositorio ():TRepositorio;
```

```
(* Crea un Repositorio vacío *)
```

```
PROCEDURE RepositorioVacio(r:TRepositorio):BOOLEAN;
```

```
(* Nos dice si un Repositorio está vacío *)
```

```
PROCEDURE RepositorioLleno(r:TRepositorio):BOOLEAN;
```

```
(* Nos dice si un Repositorio está lleno *)
```

```
PROCEDURE InsComponente(VAR r:TRepositorio; cmp:TCadena);
```

```
(* Inserta un componente software en el Repositorio *)
```

```
PROCEDURE InsAlgoritmo(VAR r:TRepositorio; cmp,alg,leng:TCadena);
```

```
(* Inserta un algoritmo en el Repositorio. Si el componente no está  
    insertado, NO lo insertar
```

```
*)
```

```
PROCEDURE BorrarComponente(VAR r:TRepositorio; cmp:TCadena);
```

```
(* Borra un componente y todos sus algoritmos del Repositorio *)
```

```
PROCEDURE BuscarComponente(r:TRepositorio; cmp:TCadena):BOOLEAN;
```

```
(* Nos dice si un componente está en el Repositorio *)
```

```
PROCEDURE PintarComponente(r:TRepositorio; cmp:TCadena);
```

```
(* Pinta por pantalla todos los algoritmos de un  
    componente del Repositorio
```

\*)

```
PROCEDURE PintarRepositorio(r:TRepositorio);  
(* Pinta por pantalla TODOS los componentes y sus  
algoritmos del Repositorio  
*)
```

```
PROCEDURE PintarLenguaje(r:TRepositorio;leng:TCadena);  
(* Pinta por pantalla TODOS los algoritmos de TODOS los componentes  
implementados en un determinado lenguaje  
*)
```

```
PROCEDURE SalvarRepositorio(r:TRepositorio; fichero:TCadena);  
(* Salva el Repositorio en un fichero de texto *)
```

```
PROCEDURE DestruirRepositorio (VAR r:TRepositorio);  
(* Destruye el Repositorio *)
```

END MRepo.

### Se valorará:

1. La corrección del programa.
2. Una buena modularización (uso de procedimientos/funciones)
3. Uso de tipos adecuados.
4. La ausencia de efectos laterales.
5. La legibilidad del código.
6. La presencia y claridad de mensajes de salida resultado de cada operación.

### Notas:

1. Se puede utilizar cualquier subrutina de los módulos estándares de TOPSPEED **EXCEPTO LA LIBRERÍA STR**
2. Es obligatorio trabajar en el directorio **C:\CMPSOFT**. Si no existe se creará. El programa principal se llamará **CMPSOFT.MOD**
3. **Para Aprobar** deberá ser correcta la definición de tipos, la modularización y funcionar **CORRECTAMENTE** las **Opciones A,B, C, D, E y X** del menú.
4. Las opciones F y G del menú servirán para obtener más de aprobado.