Ejercicios con estructuras de datos

Clasificación por inserción

Dado el tipo base adjunto de un array de tipo `TStock [1..L]`, implementa el algoritmo de clasificación por inserción según valores ascendentes del campo `clave` usando dos módulos: uno que utilice la búsqueda binaria para determinar la posición de inserción, cuya especificación viene dada por:

```
REGISTRO TArticulo
   N clave
   R precio
FINREGISTRO
```

```
N ALGORITMO BBinaria(ES TStock stock; E N desde, hasta; E TArticulo elem)
/* halla la posición de "elem", según "elem.clave", dentro del subarray
   [desde..hasta] de "stock" (hasta<=L). Si no está, devolverá cero. */
```

y otro que realice los intercambios necesarios:

```
N ALGORITMO Desplazar(ES TStock stock; E N desde, hasta)
/* desplaza los elementos de "stock" entre "desde" y "hasta" (< L) una
   posición a la derecha, dejando libre stock[desde] */
```

Clasificación por selección

Dada la estructura de datos anterior, implementa la clasificación por selección de dos formas:

a) con dos bucles anidados;
b) reutilizando el algoritmo de la búsqueda binaria del ejercicio anterior.

Clasificación por selección

Dada la estructura de datos anterior, implementa la clasificación por el método de la burbuja:

a) sin centinela;
b) con centinela.

Estructuras anidadas

```
TIPOS
   C Nombre[0..25]
   ENUM {Hardware, Software} TTipo
   REGISTRO RegistroLista
     Nombre lista[1..5000]
     Nombre sublista[1..20]
   FINREGISTRO
   REGISTRO RegCatalogo
     Nombre nombre_elem
     TTipo tipo_elem
     RegistroLista subcatalogo
   FINREGISTRO
```

```
VAR
   RegCatalogo catalogo[1..20]
   RegCatalogo un_catalogo
   RegistroLista una_lista
   Z cont
   Nombre un_nombre
```

Dadas las declaraciones anteriores, ¿cúales de las siguientes sentencias son válidas? (Asume que las variables válidas tienen ya valores definidos)

a)
```
SI (RegCatalogo.tipo_elem == Hardware) ENTONCES
   cont = cont+1
  FINSI
```
b) `catalogo[1].subcatalogo.lista[2] = un_catalogo`
c) `catalogo[5].RegistroLista = una_lista`
d) `un_catalogo.nombre_elem[2] = una_lista.lista[2]`
e) `catalogo[1].subcatalogo.lista[1,2] = un_nombre[2]`
f) `catalogo[20].lista[1] = un_nombre`
g)
```
SI (catalogo[20].tipo_elem = Software) ENTONCES
   catalogo[10].RegistroLista.sublista[3,9] = 'A'
  FINSI
```
h) `un_catalogo = catalogo[5]`
i) `un_catalogo.subcatalogo = una_lista`

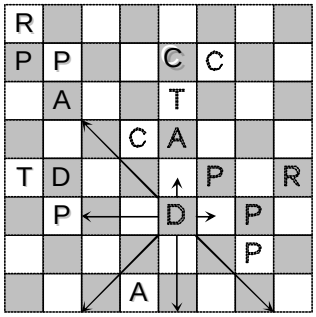
Ajedrez

La estructura de datos principal de un programa relacionado con el ajedrez es una matriz 8x8 que representa el estado de una partida en un instante dado. Cada cuadrícula del tablero es un elemento de esa matriz que puede estar vacía o contener una de las 12 posibles piezas del ajedrez. Por cada bando sólo puede haber un máximo de un *rey*, una *dama*, dos *torres*, dos *alfiles*, dos *caballos* y ocho *peones*. Durante la partida como mínimo deben estar los dos reyes, pero puede faltar cualquier otra pieza.

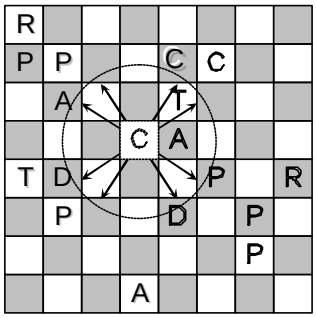
Blancas:	Negras:
R: Rey_B	R: Rey_N
D: Dama_B	D: Dama_N
T: Torre_B	T: Torre_N
A: Alfil_B	A: Alfil_N
C: Caballo_B	C: Caballo_N
P: Peón_B	P: Peón_N

Se pide diseñar los tipos adecuados necesarios y la implementación y especificación de los subalgoritmos siguientes, todos los cuales reciben como parámetro de entrada, como mínimo, la situación actual del tablero almacenada en la matriz mencionada.

a) **`AmenazaDama`**( ). Dado el estado actual de la partida, una posición (i,j) y el bando de una dama como entrada, determina si esa posición está amezada por la dama de ese color. Si la dama de ese color no está en el tablero o si no amenaza la posición (i,j), el subalgoritmo debe devolver la indicación FALSO. Si está la dama en el tablero y amenaza la posición (i,j), devolverá la indicación VERDADERO. Una dama amenaza cualquier pieza adversaria que esté en su fila, columna o diagonal siempre que no haya otra pieza entre ambas. En la figura adjunta la dama blanca amenaza un alfil y un peón negros.



b) **`AmenazaCaballo`**( ). Dado el estado actual de la partida, la posición (icab, jcab) de un caballo, y una posición (iobj, jobj) como entrada, determina si el caballo en esa posición amenaza la casilla objetivo (iobj, jobj). El caballo amenaza cualquier casilla que esté a una distancia "aproximada" de dos casillas y que no esté en su fila, columna o diagonales sin importar las piezas que puedan estar entre ambas. En la figura adjunta, el caballo blanco amenaza un alfil, un caballo y la dama del bando negro.



c) **`EvaluarFuerzas`**( ). Dado el estado actual de la partida y un determinado bando como entrada, evalúa las fuerzas de ese bando sumando los valores asociados a cada pieza según la tabla de valores adjunta. Además, se sumará un punto por cada una de las 4 casillas centrales ocupadas por un peón propio o amenazadas por un caballo propio.

Valores piezas:
D: 9 puntos
T: 5 puntos
A: 3 puntos
C: 3 puntos
P: 1 punto

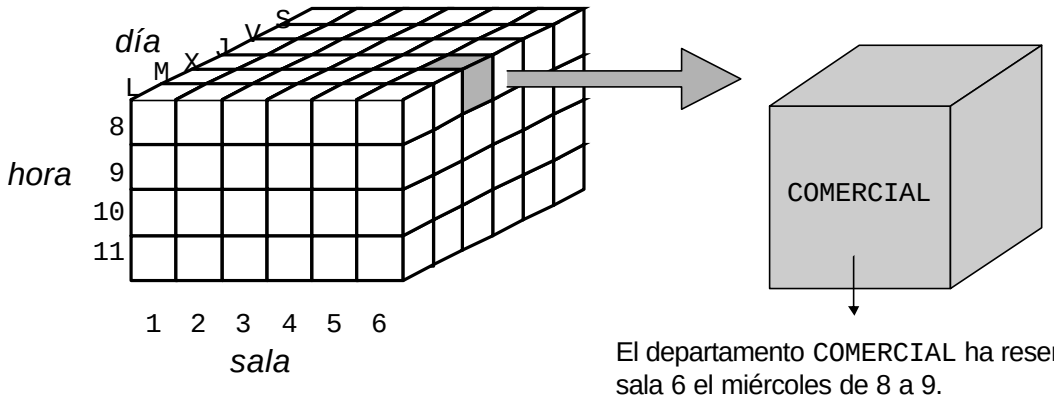
Sustitución de acrónimos

Dado un texto y la tabla de acrónimos con sus significados adjunta, diseñar e implementar un algoritmo que reciba un texto y sustituya cada acrónimo por su correspondiente significado completo.

UCP:	unidad central de proceso
UC:	unidad de control
UAL:	unidad aritmético lógica
CP:	contador de programa
PE:	palabra de estado
AC:	acumulador
MP:	memoria principal
MS:	memoria secundaria

Salas empresa

Una empresa dispone de 6 salas de reunión de uso común que pueden utilizarse en bloques de una hora desde las 8:00 a las 12:00 horas. La empresa desea desarrollar un programa para gestionar semanalmente la reserva de salas por parte de diferentes departamentos (*comercial*, *personal*, *producción* y *publicidad*). La información de las reservas debe mantenerse en una matriz (**TipoReservas**) de tres dimensiones cuyos índices son el día de la semana, la hora a reservar y la sala. El contenido de cada celda será el departamento que reserva la sala a esa hora o libre si la sala no está reservada en ese momento.



El departamento COMERCIAL ha reservado la sala 6 el miércoles de 8 a 9.

Se pide diseñar la estructura de datos adecuada e implementar los siguientes algoritmos:

1) **ALGORITMO** B **Reservar**(**ES** TipoReservas reservas; **E** TipoDpto dpto;
E TipoDia dia; **E** TipoHora hora; **E** N num_salas)

Esta función devolverá VERDADERO si hay suficientes salas libres el día y la hora indicados, en cuyo caso se cambiará adecuadamente el contenido de las celdas correspondientes, sacando por pantalla un mensaje que indique las salas asignadas:

```
"Se han reservado para su reunión las salas 1, 3, 5"
```

Si no hay suficientes salas, no se reserva ninguna, se devuelve FALSO, y se informa de los departamentos que tienen salas ya reservadas en el período en cuestión:

```
"No se efectuó ninguna reserva, los departamentos siguientes
 ya tienen salas reservadas en el período deseado:
    COMERCIAL
    PUBLICIDAD"
```

2) **ALGORITMO** B **Anular** (**ES** TipoReservas reservas; **E** TipoDpto dpto;
E TipoDia dia; **E** TipoHora hora)

Esta función devolverá VERDADERO si había alguna reserva para esa hora de ese departamento y FALSO si no la había. Además, si había reserva, se pondrán las salas como libres.

3) **ALGORITMO** **Mostrar** (**E** TipoReservas reservas)

Este procedimiento mostrará las reservas ordenadas por días, horas y salas. El resultado será similar a:

```
LUNES
   De 8 a 9 horas
      Sala 2. COMERCIAL
      ...
      Sala 6. PUBLICIDAD
   De 9 a 10 horas
      Sala 1. PRODUCCCION
      Sala 3. PERSONAL
      ...
      Sala 6. PUBLICIDAD
   ...
MARTES
   ...
MIÉRCOLES
   ...
```

S.O. multiusuario y multitarea

Un sistema operativo multiusuario y multitarea almacena la siguiente información sobre sus usuarios:

uid	nombre	ppaso	flogin			hlogin			pids				
			dia	mes	año	hh	mm	ss					
0	sistema	*******	12	04	2002	22	34	23	12333	3245	1735	34567	...
18	monica	******	07	07	2002	09	30	34	0	2351	3566	0	...
0	sistema	*****	12	04	2002	08	24	16	13444	16335	0	0	...
25	belen	******	16	07	2002	16	32	23	0	3625	2561	12667	...
39	vanesa	****	16	07	2002	17	05	51	2398	0	3352	0	...
...													...

y sus procesos:

pid	prog	dirIni	dirFin	t_ejec			estado
				hh	mm	ss	
34567	eudora			0004	12	42	Dormido
3566	netscape	2566600	3006000	0008	36	34	Espera
0							
12667	cc	0671000	1002000	0000	01	44	Activo
0							
2351	emacs	3100300	3340000	0000	23	26	Espera
3352	vi			0001	16	16	Dormido
16335	csh	0406200	0408000	0011	25	18	Activo
...							...

Se pide diseñar e implementar:
a) Las dos estructuras de datos correspondientes; **`TUsuarios`** y **`TProcesos`**;
b) Un subprograma, **`UsuariosGolosos`**, que escriba en pantalla los usuarios que estén ocupando mucha memoria (por encima de `m_umbral`) o hayan usado más de ¾ t medio de CPU.

Notas.

1. Los identificadores de usuario (*uid*) y los de procesos (*pid*) son únicos. No tener en cuenta el usuario sistema (uid=0).
2. **`dirIni`** y **`dirFin`** son direcciones de memoria.
3. Suponed que los procesos dormidos están en memoria secundaria y no ocupan memoria principal.

Profesores

Una aplicación informática de un centro universitario dispone de la siguiente información sobre una carrera:

Por cada asignatura:
a) código de asignatura (único);
b) nombre de la asignatura;
c) curso;
d) créditos;
e) dni del profesor que la imparte.

Por cada profesor:
a) nombre del profesor
b) dni;
c) fecha de ingreso en la Universidad;
d) fecha de nacimiento;
e) categoría (catedrático, titular, asociado o ayudante).

Teniendo en cuenta que un profesor puede impartir más de una asignatura, diseñar las estructuras de datos apropiadas y los siguientes subprogramas:

1) **ProfesorAsignatura**. A partir de la(s) estructura(s) de datos adecuada y el código de la asignatura, obtener el nombre del profesor que imparte la asignatura si existe la asignatura y el profesor. Devolverá FALSO si no se puede obtener el nombre por cualquier circunstancia.

2) **SueldoMedioAnual**. A partir de la(s) estructura(s) de datos adecuada calcula el sueldo medio al año de los profesores del centro. El sueldo base anual depende de la categoría:

   a) Catedrático: 2000 euros
   b) Titular: 1500 euros
   c) Asociado: 1000 euros
   d) Ayudante: 800 euros

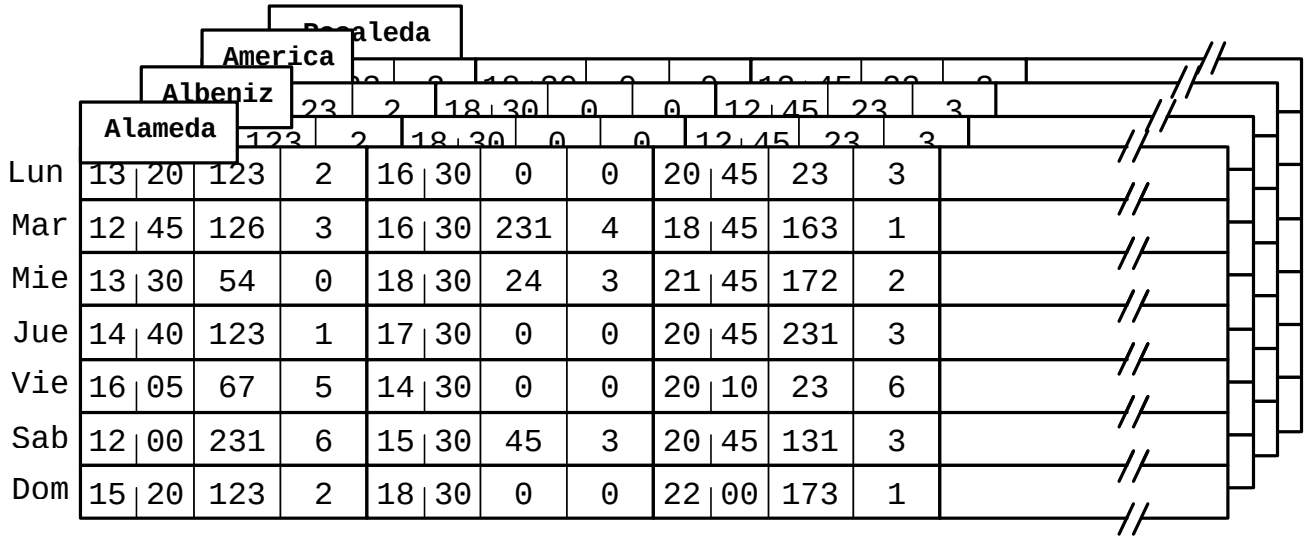
Hay que tener en cuenta que si el profesor es numerario (catedrático o titular) el sueldo incluye un complemento que se calcula en función del número de años que el profesor ha trabajado en la Universidad, cobrando 90 euros mensuales por cada cinco años completos trabajados. Suponed que la fecha actual es el 31 de diciembre de 2002.

Si el profesor es no numerario (asociado o ayudante) el sueldo no incluirá el complemento de quinquenios.

Salas de cine

Dadas las estructuras de datos adjuntas sobre las sesiones de los multicines de Málaga en la semana actual y la información sobre las distintas películas, se pide diseñar e implementar:

1. las estructuras de datos;
2. un subprograma, **Motrar_TP**, que muestre todas las películas autorizadas para todos los públicos que proyecten en todos los cines durante toda la semana.
3. un subprograma, **EscribirNoGolfas**, que muestre las proyecciones de todos los cines de las películas que acaben antes de una cierta hora para un determinado día de la semana.



codigo	titulo	prota1	prota2	publico	duracion
231	"Jarri Pota"	"Jacinto Pi"	"Pepa Guti"	TP	92
67	"Sr. anillo"	"Lolo Jones"	"Mary Perez"	TP	178
13	"Sin noticias"	"Vicky Mayo"	"Lope Cruise"	M18	94
173	"Gran familia"	"Beto Closas"	"Loli Camus"	TP	118
...	...	...	...	...	...