Problemas de examen del CUM

Junio 00 CUM

Problema 1.

Modificar el algoritmo pertinente de caminos mínimos para que un Grafo admita varios pesos por arco y se pueda elegir por cuál o cuales de estos pesos determinar qué camino es mejor en relación a otros.

Suponed para ello que ya está implementada una unidad denominada **UniPeso** que ofrece un método **ComPeso** para poder comparar pesos dependiendo del tipo al que corresponda; es decir, comparar dos pesos del tipo Real se haría llamando:

ComPeso ('real', peso1, peso2) → True si peso1 < peso2, sino False

Suponed también un máximo de 10 pesos y que sus tipos sólo pueden ser los básicos Pascal.

Si llamamos a la clase que contiene la definición de este tipo de grafo **GrafoVP, Incluye también cuál sería la definión de esta clase en Pascal.**

Problema 2.

Diseñar una estructura de datos adecuada, escribiendo la definición del *Interface* en Pascal para POO, y dibujándola de forma esquemática, según se ha ido haciendo en clase, para el siguiente caso:

Un *Holding* de empresas quiere mantener almacenada la estructura del personal que compone cada una de las empresas. Cada empresa mantiene a su personal según una jerarquía de niveles (Director como nivel 1, Subdirectores como nivel 2, Gerentes como nivel 3 dependientes de los subdirectores, etc...). Sólo hay un Director por empresa; y, dentro del *Holding*, ninguna empresa prevalece sobre otra: todas son del mismo nivel (la figura del Director tiene la misma importancia en cada una de las empresas componentes del *Holding*).

Para simplificar el problema, suponed que sólo guardaremos como información, tanto para las personas como para las empresas, sus respectivos nombres.

Implementar la operación de alta de una persona en una determinada empresa del *Holding* con un determinado nivel; por ejemplo, con la llamada al método:

Mi_Holding . Insertar_persona (Nombre-Empresa, Nombre-Persona, Nivel-Cargo)

daríamos de alta un nuevo empleado en una determinada empresa (ojo, la variable Mi_Holding es el objeto instanciación de la clase que diseñéis).

Septiembre 00 CUM

Problema 1.

Diseñar una estructura de datos adecuada, escribiendo la definición del *Interface* en Pascal para POO, y dibujándola de forma esquemática, según se ha ido haciendo en clase, para el siguiente caso:

Dado un número indeterminado de personas (un pueblo o ciudad completo, por ejemplo), queremos mantener las relaciones que existiesen entre unos y otros: p.ej., Pepote es amigo de Juancho, primo hermano de Luisito y padre de Alfonsín,....
Las relaciones podrán ser tantas y tan variadas como la realidad misma contempla.

Distintas personas pueden llamarse igual, de tal forma que habremos de añadir a cada persona un identificador que las diferencie (un campo, como p.ej. el DNI, que identifica de forma única a una persona; también podría ser otro tipo de identificación, como el número de SS, pasaporte o alguno inventado por nosotros). Este campo lo denominaremos ID y, por simplicidad, no ahondaremos en su estructura final elegida.

Definir e Implementar la estructura de datos interna en Pascal OO que pudise albergar esta información.

Implementad cómo sería el método de consulta de todas las relaciones que una persona mantiene con todas las demás almacenadas (dado Luisote, se nos devolverá todas y cada una de sus relaciones con la persona o personas implicadas)

Problema 2.

Diseñar una estructura de datos adecuada, escribiendo la definición del *Interface* en Pascal para POO, y dibujándola de forma esquemática, según se ha ido haciendo en clase, para el siguiente caso:

Queremos mantener almacenadas, para posteriormente emplearla en una aplicación *Agenda* en Internet, todas las tareas que a un número indefinido de personas se les pueda plantear.

En principio, sólo nos interesará un identificador de persona único y su nombre (incluyendo sus apellidos, claro), las tareas que desee mantener (consideradas éstas como una simple tira de caracteres de longitud suficiente), su fecha/hora en que la realice, y un indicador que nos diga si ha sido efectuada o no.

La cuestión es **implementar la estructura de datos interna en Pascal OO que sea la más eficiente posible para acceder a cualquier tarea de cualquier persona**, dada su identificación única y la fecha/hora de dicha tarea.

Implementad, así mismo**, cómo sería el método de consulta de una tarea determinada de una cierta persona, devolviéndose todos los datos sobre ella de que disponemos** (descripción textual de la tarea, fecha/hora y si se realizó o no)

Febrero 01 Problema CUM

Diseñar una estructura de datos adecuada, escribiendo su especificación y la definición del *Interface* en Pascal para POO, y dibujándola de forma esquemática, según se ha ido haciendo en clase, para el siguiente caso:

Una empresa de diseño industrial quiere mantener almacenada la estructura de las piezas que fabrica. Ésta es muy simple conceptualmente: las piezas o componentes, o bien se constituyen de otras piezas o componentes, o bien ya son indivisibles en otras siendo componentes finales.

Por ejemplo, los coches se componen de muchas piezas: ruedas, puertas, volante, motor, ...; donde, a su vez, las puertas se componen de ventanas, manivelas, abrazaderas,... las abrazaderas se componen de tornillo, tuerca y elemento tensor, siendo estos últimos finales. Pero las abrazaderas no son componentes únicos para las puertas, también lo serían para motores, enlaces musicales u otros objetos industriales (no sólo tratan coches)

Para simplificar el problema, suponed que sólo guardaremos como información, tanto para las piezas como para los componentes (finales o no), sus respectivos nombres.

Implementar la operación de consulta de piezas en los que un componente participa, devolviendo el resultado de la consulta en una lista; con la llamada al método:

Mi_ListaPiezas := Mis_Piezas . Consulta_Piezas (Nombre_Componente)

Siendo Mis_Piezas el objeto instanciación de la clase que diseñéis y Mi_ListaPiezas la instanciación de una clase (que suponemos ya está implementada en una unidad llamada Lista_de_Piezas) que mantiene listas de nombres de piezas o componentes donde se obtendrán todas las piezas encontradas.

La definición para su uso sería en Pascal 7:

```
Var
    Mi_ListaPiezas: Lista_de_Piezas;
    Mis_Piezas: <clase_diseñada>;
```

Junio 01 CUM

PROBLEMA 1.

Diseñar una estructura de datos adecuada, dibujándola de forma esquemática_que ayude a comprenderla y escribiendo sólo la definición de la Interfaz de POO, bien en Pascal 7 o bien en C++, considerando las operaciones que estiméis útiles y oportunas.

Dado un número indeterminado de compañías aéreas, se quiere mantener, en una estructura de datos adecuada en memoria principal, las relaciones que entre unas y otras existen; por ejemplo: entre Iberia y American Airlines se comparten destinos, aeropuertos, enlaces entre destinos y aeropuertos de una compañía respecto a la otra, cambios automáticos de billetes, representación en ciertos aeropuertos o países de una respecto a la otra, etc... Las relaciones pueden ser tantas y tan variadas como uno pueda imaginarse.

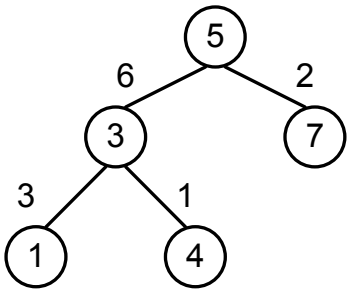
Como el problema es suficientemente abierto en sus requerimientos, cualquier decisión que se tome al respecto deberá ser suficientemente acompañada de una justificación.

Implementa, así mismo**, los métodos constructores puros que hayáis determinado y la consulta de todas las relaciones que una compañía ha pactado con otra** (téngase en cuenta que si Iberia representa a Air France en Nicaragua, no implica que Air France represente a Iberia en este mismo país o en cualquier otro).

¿Cuál es el coste de cada una de las operaciones que has implementado?

PROBLEMA 2.

Un árbol VALUADO (AV) es aquel en el que los caminos entre padres e hijos tiene asignado un número que indica la distancia entre uno y otro nodo. Por ejemplo:



En este AV, el nodo 3 es el hijo izquierdo de 5 y se encuentra situado a una distancia de 6 unidades respecto de la raíz. Cada nodo del árbol tiene la siguiente estructura :

```
Struct NodoArbol {
        Int Clave;
        Int X,Y;
}
```

La distancia entre un nodo de coordenadas (x1,y1) y su padre de coordenadas (x2,y2) viene dada por ;

distancia = $\sqrt{(x2-x1)^2+(y2-y1)^2}$

Un ABOV será un árbol binario ordenado valuado.

Se pide **diseñar un procedimiento, en Pascal 7 POO o en C++, tal que dado un ABOV muestre en pantalla un listado de las hojas del árbol y su distancia a la raíz.** Se entiende como distancia la suma de los valores de los caminos existentes entre la raíz y el nodo de que se trate. Para el ejemplo anterior, la salida de este procedimiento debe ser :

nodo 1, distancia 9
nodo 4, distancia 7
nodo 7, distancia 2

Septiembre 01 CUM

PROBLEMA 1.

Dado un número indeterminado de compañías aéreas, se quiere mantener, en una estructura de datos adecuada en memoria principal, las relaciones que entre unas y otras existen; por ejemplo: entre Iberia y American Airlines se comparten destinos, aeropuertos, enlaces entre destinos y aeropuertos de una compañía respecto a la otra, cambios automáticos de billetes, representación en ciertos aeropuertos o países de una respecto a la otra, etc... Las relaciones pueden ser tantas y tan variadas como uno pueda imaginarse.

Como el problema es suficientemente abierto en sus requerimientos, cualquier decisión que se tome al respecto deberá ser suficientemente acompañada de una justificación.

Implementa, así mismo**, los métodos constructores puros que hayáis determinado y la consulta de todas las relaciones que una compañía ha pactado con otra** (téngase en cuenta que si Iberia representa a Air France en Nicaragua, no implica que Air France represente a Iberia en este mismo país o en cualquier otro).
¿Cuál es el coste de cada una de las operaciones que has implementado?

PROBLEMA 2.

Mantener toda la información documental de forma ordenada en una empresa no es tarea sencilla para ser llevada a mano. Un gestor documental bien podría ser un objeto que recibe como entrada el título del documento, el documento mismo (su ubicación en un archivo o dirección de internet), información adicional sobre este documento (fecha, tamaño, formato, autor, ...) y las veces que se ha consultado; por ejemplo.

Si se determina que el título de un documento puede ser repetido en tantos documentos como se quiera, un identificador de éste sería necesario y aconsejable mantener.

Diseña una clase que maneje esta información de tal forma que exista la operación **Buscar** que dada una palabra cualquiera me devuelva todas las ubicaciones de archivo (su dirección, en formato literal o String) donde se encuentra esta palabra dentro de cualquier título y las veces que ha sido consultada. Para ello, Baste suponer que el documento ya está almacenado y que la estructura de datos sólo recibe como entrada la dirección/ubicación de este almacenamiento. La búsqueda debería ser todo lo eficiente que se pueda diseñar.

Implementad los métodos constructores puros que hayáis determinado y la operación de búsqueda de todas las ubicaciones en las que una palabra clave pueda encontrarse dentro de cualquier título. ¿Cuál es el coste de cada una de las operaciones que has implementado?

Febrero 02 CUM

PROBLEMA 2.

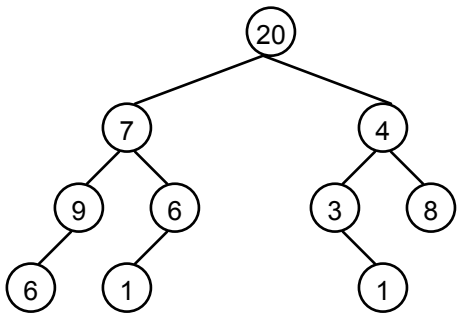
Hemos estudiado dos formas generales de hacer recorridos sobre árboles. Pero éstas no son las únicas. Existen otro tipo de recorridos que no siguen ninguno de esos esquemas. Concretamente, un algoritmo que llamaremos **recorrido_guiado**, permite realizar un recorrido sobre un árbol de forma que, en cada iteración, se selecciona el nodo más pequeño de entre todos los disponibles en ese momento, independientemente de en qué rama se encuentre. Se entiende por nodo disponible aquel nodo cuyo padre ya ha sido procesado (excluyendo el nodo raíz). Además, este tipo de algoritmos optimiza el tiempo de proceso, de forma que, cuando un árbol tiene nodos repetidos, y se llega a un nodo igual que uno anteriormente visitado, este último se olvida sin más.

Suponer además que tenemos implementada una clase **COLA** cuya operación de inserción coloca los datos en la cola ordenados de menor a mayor.

SE PIDE: Usando técnicas de POO, implementar una función que, dado un árbol binario, y empleando el algoritmo descrito anteriormente, realice un recorrido guiado por el árbol, mostrando en pantalla el contenido de cada nodo. El prototipo de la función será: **void recorrido_guiado (Arbol a);**

NOTA: Los nodos del árbol se deben procesar **una sola vez**.

Ejemplo de recorrido:



Para este árbol, el procedimiento *recorrido_guiado* debe mostrar en pantalla:

20, 4, 3, 1, 7, 6, 8, 9

¿Cuál es el coste de ejecución que emplea?