Preguntas de AAED

1.- Preguntas generales.-

• ¿Que es un TAD?

Un Tipo Abstracto de Dato es un ***conjunto de objetos***, que tiene una ***familia de operaciones*** definidas sobre dicho conjunto, del cual tenemos que tener en cuenta qué es un ***tipo de dato*** (nueva clase de objetos) y qué es una ***abstracción*** (nivel de info superior que obvia detalles irrelevantes para el tipo de datos).

• ¿Qué importancia tiene la especificación en la creación y utilización de un TAD?

Gracias a la especificación, el usuario tiene a ***toda la información*** necesaria para él del TAD: el ***Dominio*** (conjunto de valores que tomará una entidad que pertenezca a la nueva clase de objetos) y las ***operaciones*** (cómo usar los objetos y qué significado o consecuencia tiene cada operación).

• ¿Qué es la especificación sintáctica?

Determina ***cómo usar*** las operaciones, es decir, la forma en que se ha de escribir cada una, dando su nombre junto al orden y tipo de operandos y resultado.

• ¿Qué es la especificación semántica?

Expresa el ***significado*** de las operaciones, o sea, que hace y qué propiedades tiene cada una.

• ¿Qué es la implementación de un TAD?

La implementación de un TAD es la parte conocida únicamente por el diseñador y es la definición de la estructura de datos que soporta al TAD y la codificación de las operaciones de forma que exhiban las propiedades y comportamiento especificados.

•**Principio fundamental de los TAD: independencia de la representación**.

El uso de un TAD se basa en su especificación y, por tanto, también es independiente de la representación subyacente, lo que implica que, un mismo TAD, se puede implementar con distintas representaciones y los requisitos del problema o aplicación determinarán la representación más adecuada a utilizar.

• ¿Existe alguna relación entre la ocultación de información y la independencia de la representación?

Sí, ya que si queremos que sea más cómodo el uso del nuevo tipo de dato, es conveniente obviar el tipo de implementación que se ha usado(independencia de la representación), es decir, necesitamos realizar ocultamiento de la información.

• Ante la posibilidad de que no se verifiquen las precondiciones de una operación de un TAD, que decisiones de diseño puede implementar la operación?

No se afirma nada en el caso de que no se cumplan las precondiciones, puede dar un resultado indeterminado o error. Lo ideal sería que diese un error advirtiendo de esa forma el mal uso de esa operación.

• ¿Qué sucede si después de la ejecución de una determinada operación, no se cumplen las precondiciones de la misma?

NADA.

2.-Pilas.

• ¿Qué es una pila?

Secuencia de elementos de un tipo determinado en la cual se pueden añadir y eliminar elementos sólo por uno de sus extremos, llamado tope o cima. En una pila el último elemento añadido es el primero en salir de ella, por lo que también se les conoce como estructuras LIFO: Last Input First Output.

• ¿Cuando utilizamos una representación estática del TAD pila? Ventajas e inconvenientes

Usaremos la representación estática del TAD Pila cuando sepamos el número máximo de elementos que se puede almacenar en ella.

Ventajas: Ocupa menos espacio en memoria que la implementación del TAD Pila mediante celdas enlazadas ya que no requiere un puntero por elemento.

Inconvenientes: Exige fijar el tamaño máximo de la pila en tiempo de compilación, por lo que el tamaño tendrá que ser una constante y siempre el mismo para todas las pilas que creemos.

• ¿Cuándo utilizaremos una representación pseudoestática del TAD Pila? Ventajas e inconvenientes

Usaremos la representación pseudoestática del TAD Pila cuando queramos dejar al usuario que determine el número máximo de elementos que contenga la pila.

Ventajas: Nos permite fijar distintos tamaños en tiempo de ejecución (este tamaño será fijo durante todo el tiempo de vida de la pila), por lo que un tamaño inicial no determina el tamaño de todas. Ocupa menos espacio que la implementación del TAD Pila mediante celdas enlazadas ya que no requiere un puntero por elemento.

Inconvenientes: Tamaño constante durante la vida de cada pila.

• ¿Cuándo utilizaremos una representación dinámica del TAD pila? Ventajas e inconvenientes

Usaremos la representación dinámica del TAD Pila cuando no necesitemos controlar el número máximo de elementos que tenga la pila.

Ventajas: El tamaño de las pilas es variable, con lo que podemos incrementar o decrementar cuando queramos en tiempo de ejecución. Siempre ocupará el

espacio justo para almacenar los elementos que contenga la pila en cada momento.

Inconvenientes: Alto consumo de memoria debido a la utilización de un puntero por cada elemento.

• ¿Tiene algún sentido representar una pila con una lista doble enlazada?

No, ya que en una lista doblemente enlazada se usan 2 punteros (uno para el elemento siguiente y otro para el anterior) y en las pilas solo se inserta y elimina por el tope, con lo que es un gasto de memoria innecesario al valer con un único puntero.

• ¿Cuándo utilizaremos la representación circular del TAD pila?

Esta representación no existe para el TAD Pila. Tal y como se define el TAD pila, una pila es una secuencia de elementos de un tipo determinado, en la cual se pueden añadir y eliminar elementos sólo por uno de sus extremos llamado tope o cima, por lo tanto, carecería de sentido utilizar un puntero que se utilizara para apuntar a la base de la pila (primer elemento que se introduciría en ella).

• ¿Cómo se podría eliminar un elemento cualquiera de una cola/pila?

No se podría hacer, ya que tanto para pilas como para colas tenemos una especificación que ha de cumplirse, y en ninguna de las 2, se nos dice cómo hacerlo. Una forma de hacerlo sin saltarnos la especificación sería ayudaremos de una pila/cola auxiliar en la cual volcamos la que tenemos, hacer pop en el elemento que queramos eliminar y después hacer otro volcado.

• ¿Por qué en las implementaciones mediante celdas enlazadas, tanto de pilas como de colas, no existe un métodoññ llena?

No existe el método llena en estas implementaciones debido a que el tamaño es dinámico y no está definido, y por tanto, el tamaño de la estructura variará en tiempo de ejecución tanto como el usuario lo requiera.

• ¿En qué programas utilizarías el TAD Pila?

Utilizaremos el TAD Pila en aquellos casos en los que el usuario desee extraer los elementos en el orden inverso en que se insertaron y viceversa. También si se desea que todas las operaciones del TAD tengan orden constante. Algunos programas comunes en los que se usa el TAD Pila son por ejemplo líneas de texto.

3.- Colas.

• ¿Qué es una cola?

Una cola es una secuencia de elementos en la que las operaciones se realizan por los extremos:
-Las eliminaciones se realizan por el extremo llamado inició, frente o principio de la cola.
-Los nuevos elementos son añadidos por el otro extremo, llamado final de la cola.
En una cola el primer elemento añadido es el primero en salir de ella, por lo que también se les conoce como estructuras FIFO: First Input First Output.

• Ventajas e inconvenientes de una representación vectorial del TAD Cola.

-Ventajas: Ocupa un espacio fijo de memoria, el cual está representado por una estructura que contiene un vector que almacena todos los elementos, un entero que almacena el tamaño máximo del vector y otro entero que almacena el fin de la cola.

-Inconvenientes: Hay que conocer el número aproximado de datos que se van a introducir en la cola. Y además, en la operación Pop(), se habría de mover todos los elementos de la cola una posición anterior.

• Ventajas e inconvenientes de una representación circular del TAD Cola.

-Ventajas: Esta representación soluciona el inconveniente de la representación vectorial. Y además, como el principio y el fin de la cola son consecutivos el orden de las operaciones de eliminación y de inserción se reducen.

-Inconvenientes: Es distinguir entre una cola llena y una cola vacía, ya que la posición de los índices de inicio y fin sería la misma en ambos casos. Las soluciones propuestas serían añadir un indicador de si está o no vacía la cola o añadir una posición más al vector, la cual siempre estaría vacía: Si inicio sigue a fin, la cola está vacía; si entre inicio y fin hay una posición vacía, cola llena y si inicio = fin solo hay un elemento.

• Eres un diseñador del TAD Cola y un usuario te solicita una operación de acceso al n-ésimo elemento de la misma. ¿Incluirías esta operación en el TAD?

Si el usuario es mi cliente, sí (el cliente manda). Por norma general, hay que ser reacios a modificar el concepto de Cola, pero por otro lado, sería una operación cómoda que, para todas las implementaciones conocidas (basadas en vectores o listas enlazadas), se podría implementar sin un coste adicional para el resto de operaciones (esa operación sería ineficiente -coste lineal, pero las restantes mantendrían su coste). Habría que informar al cliente sobre sus perjuicios (una cola no está preparada para acceder al n-ésimo elemento de manera eficiente), pero podría añadirse si se requiere por alguna razón (la clase cola de la STL, por ejemplo, tiene iteradores de acceso para poder recorrer la cola).

No, ya que esta operación no es una operación del TAD Cola (No está definida en la especificación del TAD Cola y esta no se puede modificar). Se tendría que crear otro TAD correspondiente a las necesidades del usuario.

•¿Por qué en la implementación vectorial circular de las colas, existe una posición que siempre está vacía en el vector?

Existe una posición que siempre deba estar vacía para que se pueda distinguir cuando está la cola vacía y cuando está llena. Estará vacía cuando inicio siga a fin, y llena, cuando entre ambas posiciones quede una libre, la que hemos sacrificado.

• La representación de una cola mediante estructuras enlazadas, ¿por qué se representa usando dos punteros a cada extremo de la misma?

La representación de una cola mediante estructuras enlazadas podría realizarse utilizando un solo puntero a uno de los extremos de la misma, o mediante dos punteros, uno a cada extremo, como en este caso. Una posible representación, utilizando tan solo un puntero, sería apuntar al fin de la misma y enlazar el último elemento con el primero, de forma que se accedería al fin de la cola en coste O(1), y al inicio en coste O(2). Otra posibilidad, acorde con los accesos al TAD cola, es representarla mediante dos punteros que apunten al inicio y fin de la cola. Su ventaja es que el acceso a ambos extremos de la cola se consigue en coste O(1), mejorando a la representación con un único puntero. Su inconveniente es que requiere dos punteros en vez de uno para representar la misma (más memoria)

• ¿Para qué se utiliza el nodo cabecera en el TAD Cola, en su implementación con celdas enlazadas? //PREGUNTA de C(anticuada)

El nodo cabecera no se utiliza en el TAD cola, es innecesario dado que ésta surge para resolver un problema de independencia de la representación en el TAD lista, cuando definimos la posición en esta.

• ¿Con la implementación de colas mediante un vector circular de tamaño n cuántos elementos pueden almacenarse?

Se podrían almacenar n-1 elementos, ya que una de las posiciones del vector se debe reservar para poder distinguir cuando está llena o vacía la cola.

• Con la representación de colas mediante una estructura enlazada, con puntero al final y circular. ¿Cuántos elementos pueden almacenarse?

Pueden almacenarse todos los elementos que se quiera, ya que la representación de colas mediante una estructura enlazada es dinámica y, por tanto, no hay un límite de elementos a almacenar.

• Comenta la siguiente afirmación: El TAD Cola Circular es un TAD representado mediante un vector circular en el que el número de elementos a insertar como máximo es n-1.

El TAD Cola Circular no existe, pero sí existe una representación circular del TAD Cola, en el que se pueden almacenar n-1 elementos, debido al uso de una posición de esta como indicador de cola llena/vacía.

• ¿En qué programas utilizarías el TAD Cola?

Utilizaremos el TAD Cola en aquellos casos en los que el usuario desee extraer los elementos en el mismo orden en que se insertaron. También si se desea que todas las operaciones del TAD tengan orden constante.

4. Listas

• ¿Qué es una lista?

Una lista es una secuencia de elementos de un tipo determinado. Se representa de la forma
L = (a1, a2, ..., an), donde n ≥ 0 y longitud = n.
Si n = 0 es una lista vacía.

Posición: Lugar que ocupa un elemento en la lista. Los elementos están ordenados de forma lineal según las posiciones que ocupan. Todos los elementos salvo el primero tiene un único predecesor, y todos los elementos salvo el último tienen un único sucesor.

Posición Final: Posición siguiente a la del último elemento. Esta posición no está ocupada nunca por ningún elemento.

• ¿En una representación vectorial de una lista, como representamos la posición fin?

La posición fin en una lista está representada mediante la longitud.

• Ventajas e inconvenientes de la implementación vectorial del TAD Lista.

Ventajas: Las posiciones podrían representarse por enteros en lugar de un puntero a la struct nodo(como en las representaciones enlazadas) y además, se puede permitir el acceso directo a un elemento en concreto por dicho entero, es decir, por la clave.

Inconvenientes: Las operaciones insertar() y eliminar() pueden ser muy costosas, ya que habría que mover el resto de elementos una posición, y otro inconveniente, sería especificar el tamaño máximo de la lista.

• Ventajas e inconvenientes de la implementación dinámica del TAD Lista

Ventajas: La representación con punteros utiliza solo el espacio necesario para almacenar los elementos que tiene la lista en cada momento, por lo que no se desaprovecha memoria.

Inconvenientes: Si la lista es demasiado pequeña, puede ser que se “desaproveche” más espacio para albergar al puntero de cada nodo que a la propia lista.

• ¿Por qué surge el nodo cabecera en el TAD Lista?

El nodo cabecera surge porque en la representación del TAD Lista mediante una estructura enlazadas sin nodo cabecera, no se pueden pasar por referencia a las operaciones insertar() y eliminar() las posiciones devueltas por anterior(), fin(), primera() y última() lo que implica que estamos incumpliendo la especificación.

Con el nodo cabecera se consigue reducir el orden de complejidad de las operaciones insertar y eliminar, de O(n) a O(1), ya que con la cabecera no es necesario recorrer la lista en busca de la posición anterior a la que se

desea insertar o eliminar. Las operaciones buscar(), anterior() y fin() siguen siendo O(n).

Porque si no existiese el nodo cabecera, nos estaríamos saltando el principio de la independencia de la representación, así pues, utilizando el nodo cabecera estaremos cumpliendo la especificación del TAD en las funciones insertar y eliminar para que todos los elementos tengan predecesor y así ninguno tenga que tratarse de forma diferente.

• ¿Cuando se utiliza un TAD Lista Circular?

Cuando no hay definido un primer y un último elemento de la lista, es decir, cuando no se sabe cuales son los extremos de la lista, por lo que todos los nodos que posee la lista están seguidos unos de otros, y por lo tanto se puede acceder a todos los nodos a partir de uno cualquiera. También, hay que tener en cuenta que es innecesario el nodo cabecera y la operación fin().

• ¿Diferencia entre TAD Lista Circular y TAD Lista?

La diferencia principal entre éstos dos TADs es que, en la Lista Circular, a diferencia de la Lista, todos los elementos tienen estrictamente un sucesor y un predecesor.
Una Lista Circular no posee posición primera() ni posición fin(), sólo posee una posición llamada inipos() que devuelve una posición para tomarla como referencia. También cabe decir que el TAD Lista Circular, a diferencia del TAD Lista, no posee cabecera ya que es innecesaria.

• ¿Para que se utiliza una implementación mediante estructura enlazada doble?

Se utiliza para reducir el orden de las operaciones anterior y fin, de O(n) a O(1), ya que en esta implementación, existe, además del puntero a la posición siguiente, un puntero a la anterior. La operación buscar() sigue siendo la única operación con O(n) y no se puede mejorar.

• ¿Cuando es conveniente utilizar una implementación mediante estructura enlazada doble?

Es conveniente utilizar esta implementación cuando se vayan a utilizar mucho las operaciones anterior() y fin(), para así evitar el mayor coste de estas.

• En la lista Circular, ¿es posible implementar alguna operación de orden logarítmico?

No, todas las operaciones son de orden 1 o de orden n, ya que o se referencia al elemento al cual apunta el puntero, o a alguno contiguo, o para las operaciones de orden n, se ha de recorrer toda la lista circular hasta hallar el elemento que se desea.

• ¿Qué condición tiene que cumplir una lista para que la búsqueda se logarítmica? ¿Y orden cuadrático?

Una lista no podrá tener orden logarítmico en la búsqueda. Para que fuese logarítmico, debería implementarse la búsqueda siguiendo un esquema de búsqueda dicotómica, la cual solo se usa en vectores, no en listas. Y una búsqueda de orden cuadrática no tendría sentido ya que el orden normal de búsqueda es O(n), el orden cuadrático sería empeorar el orden.

• ¿Por qué en la especificación del TAD Lista, en la operación anterior, las precondiciones son L = (a1, a2, . . . , an) 2 ≤ p ≤ n + 1?

El hecho de que p debe ser mayor o igual que 2 y menor o igual que n+1, viene dado porque la primera posición no tiene anterior y la posición fin, aunque no tenga elementos, si posee una posición anterior.

• ¿Por qué en la especificación del TAD Lista, en la operación siguiente, las precondiciones son L = (a1, a2, . . . , an) 1 ≤ p ≤ n?

El hecho de que p deba ser mayor o igual que 1 y menor o igual que n, viene dado porque la primera posición tiene siguiente y el último elemento no tiene siguiente (En realidad si tiene siguiente, pero es la posición fin, la cual no posee elementos).

• ¿Por qué en la especificación del TAD Lista Circular, en las operaciones anterior y siguiente, las precondiciones son L = (a1, a2, . . . , an, a1) 1 ≤ p ≤ n?

El hecho de que p debe ser mayor o igual que 1 y menor o igual que n, viene dado porque todas y cada una de las posiciones de la lista circular, poseen una posición anterior y una posición siguiente.

• ¿Qué pasaría si insertas o eliminas en la primera posición en una implementación de lista mediante estructura enlazada sin cabecera?

Nada, solamente se vuelven a redistribuir las posiciones, es decir, si eliminamos, el elemento que está en la posición 2 se convierte en la 1, y si inserto, el nuevo elemento toma la posición 1, y el que anteriormente estaba en la 1 se convierte en la 2 y así sucesivamente.

• Representando el TAD Lista mediante un vector ¿es posible evitar el coste O(n) para inserciones y borrados en los extremos?

Sí, implementándolo mediante un vector circular, al igual que se hace con la representación circular del TAD cola. //requiere explicación
Migue: esto hay k hablarlo

• ¿En qué programas utilizarías el TAD Lista?

Utilizaremos el TAD Lista en aquellos en casos en los que se desee insertar y eliminar elementos en posiciones concretas de la estructura.