Preguntas teoría AAED

1. ¿Qué es un TAD?

Un *tipo abstracto de datos* es una colección de datos con las mismas características y especificación de sus operaciones y propiedades, del cual tenemos que tener en cuenta qué es un tipo de datos (nueva clase de objetos) y una abstracción (nivel de información superior que obvia los detalles irrelevantes para el TAD).

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

Gracias a la especificación el usuario tiene acceso a toda la información que necesita del TAD: dominio (conjunto de valores que puede tomar una entidad de la nueva clase) y las operaciones (cómo usar los objetos y qué significado o consecuencia tiene cada operación).

Además, la especificación actúa como contrato entre el usuario y el diseñador. El usuario puede utilizar el TAD bajo las condiciones de la especificación y el diseñador está obligado a implementarlo de modo que asegure el comportamiento especificado.

3. ¿Qué es la especificación sintáctica?

Indica cómo usar las operaciones, es decir, la forma en la que se ha de escribir cada una de ellas. (Dando el nombre, junto al orden y el tipo de operandos y resultado).

4. ¿Qué es la especificación semántica?

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

5. ¿Qué es la implementación de un TAD?

Es la parte conocida únicamente por el diseñador. Compuesta por la estructura de datos que representa los elementos del dominio del TAD, y la implementación de los algoritmos de tratamiento de dicha estructura, que realizan las operaciones del TAD tal y como se ha especificado.

6. Principio fundamental de los TADs: 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/aplicación determinarán qué representación es la más adecuada.

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

Sí, ya que, si se quiere facilitar el uso del TAD, es conveniente obviar el tipo de implementación que se ha usado (independencia de la representación), es decir, necesitamos realizar ocultamiento de la información.

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

Nada, las precondiciones deben cumplirse antes de que se realice la operación, y no deben por qué cumplirse después de ésta.

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

En el caso de que no se cumplan las precondiciones de una operación puede dar un resultado indeterminado o un error. Lo ideal sería que diese un error advirtiéndonos del mal uso de esa operación. (assert).

10. ¿Qué es una Pila?

Es una secuencia de elementos en la que todas las operaciones se realizan por el mismo extremo, denominado *tope o cima*. En una pila el último elemento en añadir es el primero en salir (Estructuras *LIFO: Last Input First Output*).

11. ¿Cuándo utilizamos una representación estática del TAD Pila? Ventajas e inconvenientes.

Cuando conocemos el número máximo de elementos que se van a almacenar en ella.

Ventajas: Ocupa menos espacio en memoria que la representación mediante celdas enlazadas, ya que, no requiere un puntero en cada elemento.

Inconvenientes: Exige fijar el tamaño máximo de la pila en tiempo de compilación, por lo que será constante para todas las pilas que se creen.

12. ¿Cuándo utilizamos una representación pseudoestática del TAD Pila? Ventajas e inconvenientes.

Cuando queremos que el usuario determine cuál será el número máximo de elementos que contenga la pila.

Ventajas: Nos permite fijar distintos tamaños en tiempo de ejecución, por lo que un tamaño inicial no determina el tamaño de todos. Ocupa menos espacio que la representación mediante celdas enlazadas, ya que, no se requiere un puntero en cada elemento.

Inconvenientes: El tamaño es constante durante la vida de cada pila.

13. ¿Cuándo utilizamos una representación dinámica del TAD Pila? Ventajas e inconvenientes.

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 incrementarlo o decrementarlo cuando queramos en tiempo de ejecución. Siempre ocupará el espacio justo para almacenar los elementos que contenga la pila en cada instante.

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

14. ¿Tiene algún sentido representar una *pila* con una *lista* doblemente enlazada?

No, ya que, en una lista doblemente enlazada se usan dos punteros (uno para el siguiente elemento, y otro para el anterior), y en las pilas solo se inserta y se elimina por el tope, con lo que es un gasto de memoria innecesario al poderse implementar con un solo puntero. En la pila solo necesitamos apuntar al siguiente elemento, y no al anterior.

15. ¿Cuándo utilizaremos la representación circular del TAD pila?

Esta representación no existe para el TAD pila. Tal y como se define el TAD pila, carecería de sentido usar un puntero que apuntase a la base de la pila (primer elemento introducido).

16. ¿Cómo se podría almacenar un elemento cualquiera de una pila/cola?

No se podría, ya que tanto pilas como colas tienen una especificación que han de cumplirse y en esta no se indica como poder realizar dicha operación. Una forma de hacerlo sin incumplir la especificación es mediante el uso de una pila/cola auxiliar, en la cual volcaremos la actual, y añadir el nuevo elemento en el lugar que nos interese.

17. ¿Por qué en las implementaciones mediante celdas enlazadas, tanto de pila como de cola no existe un método *llena()*?

No existe dicho método ya que el tamaño es dinámico y no está definido, por tanto, el tamaño de la estructura puede variar en tiempo de ejecución tanto como el usuario lo requiera.

18. ¿En qué programas utilizarías el TAD Pila?

En aquellos casos en los que el usuario desee extraer los elementos en orden opuesto en el que se insertaron, y viceversa. También en aquellos casos en los que se desee que las operaciones del TAD tengan constante(O(1)). Por ejemplo, en algunos programas de *líneas de texto.*

19. ¿Qué es una cola?

Es una secuencia de elementos en la que las operaciones se realizan por un extremo, denominado *frente/inicio,* y los nuevos elementos se insertan por el otro extremo, llamado *fondo/final.* El primer elemento añadido es el primero en salir. (Estructura FIFO: *First Input First Output*).

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

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

Inconvenientes: Hay que conocer el máximo número de datos a insertar en la cola y, además, en la operación *pop()* se han de mover todos los elementos de la cola a la posición anterior.

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

Ventajas: Esta representación soluciona los inconvenientes de la representación vectorial y, además, como el principio y el fin son consecutivos el orden de las operaciones de *inserción() y eliminación()* se reduce.

Inconvenientes: Es más complejo distinguir entre una cola llena y una vacía, ya que la posición de los índices de inicio y fin sería la misma en ambos casos.

Las soluciones propuestas son:

- Un indicador, para saber si la cola está llena o no, o si está vacía o no.
- Añadir una posición más al vector la cual siempre estaría vacía: si sigue a fin, la cola está vacía; si entre inicio y fin hay una pos, está llena y si inicio=fin, solo hay un elemento.

22. 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?

No, ya que esta operación no pertenece al TAD cola; no se encuentra incluida en su especificación. Se tendría que crear un nuevo TAD cuya especificación cumpla con las condiciones requeridas por el usuario.

23. ¿Por qué en la implementación vectorial circular de las colas existe una posición del vector que siempre está vacía?

Para poder distinguir cuándo se encuentra la cola vacía/llena. Estará vacía si inicio sigue a fin, la cola está vacía; si entre inicio y fin hay una pos, está llena y si inicio=fin, solo hay un elemento.

24. La representación de una cola mediante celdas 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. Una posible representación usando un solo puntero, que apunte al fin de la cola, y enlazar el ultimo con el primero, de manera que se accedería al fin en *O(1) y* al inicio con coste *O(2)*. Otra posible, acorde con los accesos al TAD cola es representarla mediante dos punteros que apunten al inicio y al fin de la cola.

Su ventaja es que el acceso a ambos extremos de la cola se consigue un coste de O(1), su inconveniente es que requiere dos puntero para representar la cola, lo cual ocupa más memoria.

25. ¿Para qué se utiliza el nodo cabecera en el TAD Cola, en su implementación con celdas enlazadas?

El nodo cabecera no se utiliza en TAD cola, es innecesario dado que éste surge para resolver el problema de independencia de la representación del TAD lista, cuándo definimos la posición de éste.

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

Pueden almacenarse tantos elementos como quieras, ya que la representación de colas mediante celdas enlazadas es dinámica, y por tanto, no hay límite en el número de elementos que se puedan almacenar.

27. 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, existe una implementación circular del TAD cola en el que se pueden almacenar N-1 elementos, debido al uso de una posición de ésta como indicador de cola llena/vacía.

28. ¿En qué programas utilizaría el TAD cola?

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

29. ¿Qué es una Lista?

Es una secuencia de elementos del mismo tipo, cuya longitud, $n \geq 0$, es el número de elementos que contiene. Si n = 0, es decir, si la lista no tiene elementos, se denomina lista vacía. Los elementos están ordenados linealmente según la posición que ocupan cada uno de ellos dentro de la lista. Todos los elementos menos el primero tienen un único predecesor, y todos los elementos menos el último tienen un único sucesor. Existe una posición especial *fin()*, que sigue al último elemento y que nunca está ocupada por elemento alguno. Es posible acceder, insertar y suprimir elementos de cualquier pos de la lista.

30. En una representación vectorial de una lista, ¿cómo se representa la posición *fin()*?

Se representa mediante la longitud de la lista en un momento dado.

31. ¿Cuándo se utiliza el TAD circular?

Cuando no hay definido un primer y ultimo elemento de la lista, es decir, cuando no se sabe cuáles son los extremos de la lista, por lo que todos los nodos que posee la lista están seguidos unos de otros. Esto permite acceder a todos los nodos que posee la lista a partir de uno cualquiera, ya que están seguidos unos de otros.

También, hay que tener en cuenta que es innecesario el nodo cabecera y la posición fin();

32. Ventajas e inconvenientes de la representación vectorial del TAD lista

Ventajas: Las posiciones podrían representarse como enteros en lugar de un puntero al nodo, como en las representaciones enlazadas, y además se puede admitir el acceso directo a un elemento en concreto por dicho entero(el índice).

Inconvenientes: Las op. Insertar () y eliminar() son muy costosos ya que habrá que mover el resto de elementos una posición y otro inconveniente sería especificar el tamaño de la lista.

33. 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 desaprovecha memoria.

Inconvenientes: Si la lista es demasiado pequeña puede ser que emplee más espacio en albergar al puntero de cada nodo que la propia lista.

34. ¿Por qué surge el nodo cabecera en el TAD lista?

El nodo cabecera surge en la representación del TAD lista mediante una estructura enlazada sin nodo cabecera, no se puede pasar por referencia a las op. Insertar() y eliminar() las posiciones devueltas por anterior(), fin(), y primera() lo que significa que estamos incumpliendo la especificación. Con el nodo cabecera, se consigue reducir el orden de complejidad de las op. 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). Además si no existiese el nodo cabecera, no se cumpliría el principio de independencia de la representación, así pues utilizando el nodo cabecera estamos 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.

35. Diferencia entre TAD lista circular y TAD lista.

La diferencia principal entre estos 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 pos fin() solo posee una 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 nodo cabecera ya que es innecesario.

36. ¿Para qué se utiliza una implementación mediante estructura enlazable doble?

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

37. ¿Cuándo es conveniente utilizar una implementación mediante estructura enlazada doble?

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

38. En una 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 desea.

39. ¿Qué condición debe cumplir una lista para que la búsqueda sea orden logarítmico? ¿Y de orden cuadrático?

Una lista no podría tener orden logarítmico. Para que fuese logarítmico debería implementarse la búsqueda siguiendo un esquema de búsqueda siguiendo un esquema de búsqueda dicotómica, la cual se usa en vectores, no en listas.

Una búsqueda de orden cuadrático no tendría sentido ya que el orden normal de búsqueda es O(n), el orden cuadrático empeoraría el orden.

40. ¿Por qué en la especificación del TAD lista, en la operación anterior(), las precondiciones son L=(a1,a2,a3, … ,an) 2 <= p <= n+1?

El hecho de que p tenga que ser mayor o igual que 2 y menor igual que n + 1 viene dado por que la primera posición no tiene anterior y la posición fin(), aunque no tenga elemento, posee una posición anterior.

41. ¿Por qué en la especificación del TAD lista, en la op. siguiente(), las precondiciones son L=(a1,a2,a3, … ,an) 1 <= p <= n?

El hecho de que p debe ser mayor igual que uno y menor igual que n viene dado por que la primera posición tiene siguiente y el último, aunque tiene siguiente, no se incluye ya que es la posición fin(), que no tiene elemento.

42. ¿Por qué en la especificación del TAD lista circular, en las operaciones anterior() y siguiente(), las precondiciones son L=(a1,a2,a3, … ,an) 1 <= p <= n.

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

43. ¿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 pasa a la 1 y si se inserta, el nuevo elemento toma la posición 1 y el que estaba en la 1 pasa a la 2 y así sucesivamente.

44. Representando el TAD lista mediante un vector ¿es posible evitar el coste O(n) para inserciones y borrado en los extremos?

Esto sería posible si se implementa empleando un vector circular, al igual ocurre en la implementación circular del TAD cola.

45. ¿En que programas utilizarías el TAD lista?

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