TSP - PROGRAMACION II

Tipos Abstractos de Datos

TAD

TIPOS ABSTRACTOS DE DATOS

Abstracción: consiste en ocultar las características de un objeto y obviarlas, de manera que solamente utilizamos el nombre del objeto en nuestro programa.

Abstracción funcional:

- Crear procedimientos y funciones e invocarlos mediante un nombre donde se destaca que hace la función y se ignora cómo lo hace.

Abstracción de datos:

- Tipo de datos: proporcionado por los lenguajes de alto nivel. La representación usada es invisible al programador, al cual solo se le permite ver las operaciones predefinidas para cada tipo.
- Tipos definidos: por el programador que posibilitan la definición de valores de datos más cercanos al problema que se pretende resolver.

TIPOS ABSTRACTOS DE DATOS vs ORIENTACIÓN A OBJETOS

POO: Como contenedores

- En general, no se sabe cuántos objetos se van a necesitar para resolver un problema en particular, cuánto durarán o cómo almacenar esos objetos. ¿Cómo puede saber cuánto espacio crear si esa información no se conoce hasta el tiempo de ejecución?
- Las Arreglos y Matrices poseen este problema.
- La solución a la mayoría de estos problemas se resuelve creando otro objeto contenedor que tiene referencia a objetos internos.
- El nuevo tipo de objeto se encargará de “acomodar” todos los objetos que se coloquen dentro de él, sin saber cuántos serán. Simplemente crea un objeto contenedor y deja que se encargue de los detalles.
- Los contenedores proporcionan diferentes tipos de interfaces y comportamiento externo, además de permitir incluir operaciones que tienen diferentes eficiencias para ciertas operaciones.

Ejemplo con Arreglo

- Se puede crear una variable simple de tipo numérico y asignarle un valor.

```
int numero = 5;
```

- Cuando necesito trabajar con una gran cantidad de valores de un mismo tipo, se puede recurrir a un nuevo tipo denominado Arreglo.
- El Arreglo permite “agrupar” los valores de tipo numérico en una sola variable.

```
int[] arreglo = new int[10];

arreglo[0] = 5;
```

- Además, permite manipular los valores con nuevas operaciones, como la asignación o lecturas a través de un índice.
- También agrega nuevas operaciones, como la capacidad de obtener la longitud del arreglo.

```
System.out.println(arreglo.length);
```

TIPOS ABSTRACTOS DE DATOS vs ORIENTACIÓN A OBJETOS

TAD

- Al igual que los Arreglos, se puede crear nuevos “Tipos de Datos” con capacidades diferentes, que posean operaciones especiales y que oculten la implementación de las mismas a los Usuarios que lo quieren utilizar.
- Un Tipo de dato abstracto es un conjunto de datos u objetos al cual se le asocian operaciones.
- El TDA provee de una interfaz con la cual es posible realizar las operaciones permitidas, abstrayéndose de la manera en cómo están implementadas dichas operaciones.
- Los objetos permiten encapsular valores, métodos y funciones que pueden ser usados por otras partes de un sistema mediante una clara definición o interfaz que define a los mismos.

TIPOS ABSTRACTOS DE DATOS vs ORIENTACIÓN A OBJETOS

Ejemplo de TAD

- Se define un nuevo “Tipo de Dato” que posee las mismas capacidades que un Arreglo, pero que posee la capacidad de crecer dinámicamente a medida que nuevos valores sean asignados.
- El nuevo tipo contendrá operaciones para agregar, actualizar y eliminar valores, así también como operaciones para recorrerlo y retorna la longitud del mismo.

```
MyTad miTad = new MyTad();

miTad.agregar(5);

miTad.length();

miTad.recorrer();
```

TSP - PROGRAMACION II

ESTRUCTURAS DE DATOS LINEALES

Listas - Pilas - Colas

ESTRUCTURA DE DATOS LINEALES

- Listas
  - Simples
  - Doblemente enlazadas
  - Circulares
    - Simples
    - Doblemente enlazadas
- Pilas
- Colas

TSP - PROGRAMACION II

LISTAS

ESTRUCTURA DE DATOS LINEALES - Listas Simples enlazadas

- Lista enlazada de nodos, donde cada nodo tiene un único campo de enlace. Una variable de referencia contiene una referencia al primer nodo, cada nodo (excepto el último) enlaza con el nodo siguiente, y el enlace del último nodo contiene NULL para indicar el final de la lista.

```
record Node {
   data // El dato almacenado en el nodo
   next // Una referencia al nodo siguiente, nulo para el último nodo
}

record List {
    Node PrimerNodo   // Apunta al primer nodo de la lista; nulo para la lista vacía
}

node := list.PrimerNodo
while node not null {
    node := node.next
}
```

ESTRUCTURA DE DATOS LINEALES - Listas Doblemente enlazadas

- Un tipo de lista enlazada de dos vías. Cada nodo tiene dos enlaces: uno apunta al nodo anterior, o apunta al valor NULL si es el primer nodo; y otro que apunta al nodo siguiente, o apunta al valor NULL si es el último nodo.

```
record Node {
    data // El dato almacenado en el nodo
    next // Una referencia al nodo siguiente, nulo para el último nodo
    prev // Una referencia al nodo anterior, nulo para el primer nodo
}
```

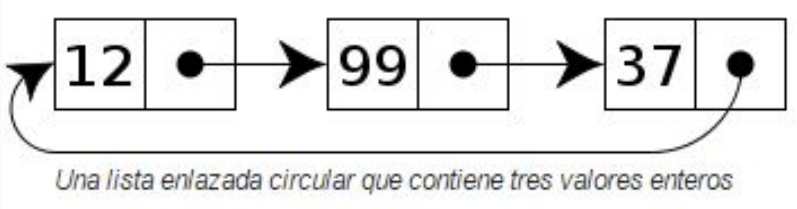
```
record List {
    Node firstNode   // apunta al primer nodo de la lista; nulo para la lista vacía
    Node lastNode    // apunta al último nodo de la lista; nulo para la lista vacía
}
```

```
node := list.firstNode
while node ≠ null
    <do something with node.data>
    node := node.next
```

```
node := list.lastNode
while node ≠ null
    <do something with node.data>
    node := node.prev
```

ESTRUCTURA DE DATOS LINEALES - Listas enlazadas circulares

- En una lista enlazada circular, el primer y el último nodo están unidos juntos.
- Esto se puede hacer tanto para listas enlazadas simples como para las doblemente enlazadas.
- Para recorrer una lista enlazada circular podemos empezar por cualquier nodo y seguir la lista en cualquier dirección hasta que se regrese hasta el nodo original.
- Este tipo de listas es el más usado para visitar todos los nodos de una lista a partir de uno dado.



Una lista enlazada circular que contiene tres valores enteros

ESTRUCTURA DE DATOS LINEALES - Listas enlazadas simples circulares

- Cada nodo tiene un enlace, similar al de las listas enlazadas simples, excepto que el siguiente nodo del último apunta al primero.
- Como en una lista enlazada simple, los nuevos nodos pueden ser insertados después de uno que ya tengamos referenciado.
- Es usual quedarse con una referencia solamente al último elemento en una lista enlazada circular simple, esto nos permite rápidas inserciones al principio, y también permite accesos al primer nodo desde el puntero del último nodo.

```
node := someNode
do
    do something with node.value
    node := node.next
while node != someNode
```

```
node := someNode
do
    do something with node.value
    node := node.prev
while node := someNode
```

ESTRUCTURA DE DATOS LINEALES - Listas enlazadas doblemente circulares

- En una lista enlazada doblemente circular, cada nodo tiene dos enlaces, similares a los de la lista doblemente enlazada, excepto que el enlace anterior del primer nodo apunta al último y el enlace siguiente del último nodo, apunta al primero.
- Como en una lista doblemente enlazada, las inserciones y eliminaciones pueden ser hechas desde cualquier punto con acceso a algún nodo cercano.
- Un puntero de acceso externo puede establecer el nodo apuntado que está en la cabeza o al nodo cola, y así mantener el orden tan bien como en una lista doblemente enlazada.

TSP - PROGRAMACION II

PILAS

ESTRUCTURA DE DATOS LINEALES - Pilas

Pilas

- Estructura de datos lineal donde los elementos pueden ser añadidos o removidos solo por un extremo.
- LIFO -> Last In - First Out

Operaciones

- Push (insertar/apilar): Agrega elementos a la pila en el extremo Tope.
- Pop (remover/desapilar): Remueve el elemento de la pila que se encuentra en el extremo llamado Tope.
- Vacía: Indica si la pila contiene o no elementos.
- Llena: Indica si es posible o no agregar nuevos elementos a la pila.

TSP - PROGRAMACION II

COLAS

ESTRUCTURA DE DATOS LINEALES - Colas

Colas

- Es una lista lineal de elementos en la que las operaciones de insertar y eliminar se realizan en diferentes extremos de la cola
- FIFO -> First In - First Out -> el primer elemento en entrar es el primer elemento en salir.

Operaciones

- Enqueue (insertar): Almacena al final de la cola el elemento que se recibe como parámetro.
- Dequeue (eliminar): Saca de la cola el elemento que se encuentra al frente.
- Vacía: Indica si la cola contiene o no elementos.
- Llena: Indica si es posible o no agregar nuevos elementos a la cola.