

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

- Objetos: Son TDA a los que se añade propiedades de reutilización y de compartición de código.
- La Programación Orientada a Objetos o OOP es la evolución de la Programación funcional.
- 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.

TSP - PROGRAMACION II

ESTRUCTURAS DE DATOS LINEALES

Listas - Pilas - Colas

ESTRUCTURA DE DATOS LINEALES

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

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.



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.

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.

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.