

UNIVERSIDAD AUTONOMA DEL ESTADO DE MEXICO

CENTRO UNIVERSITARIO UAEM ATLACOMULCO

ESTRUCTURAS DE DATOS

ING. GREGORIO GARCIA ESTRADA

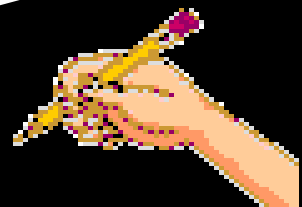
INTEGRANTES:

ALONSO GOMEZ SANDOVAL

TERESA PEREZ GONZALEZ

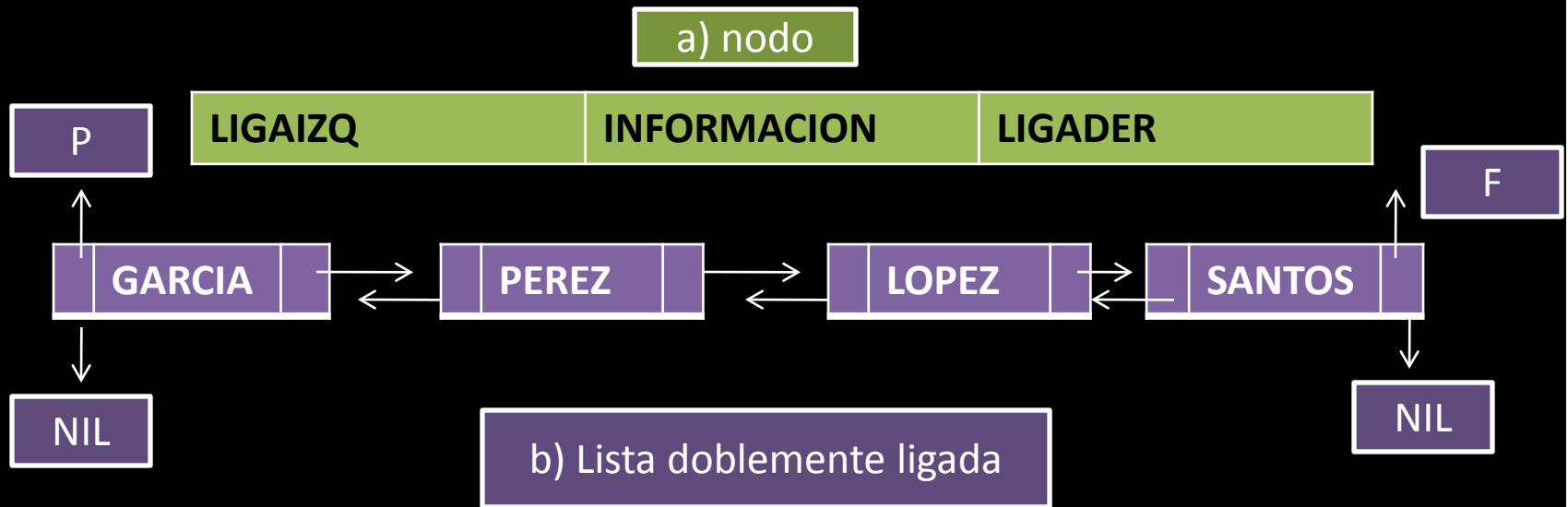
ICO 14

LISTAS DOBLEMENTE ENLAZADAS



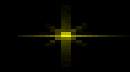
✦ QUE ES UNA LISTA DOBLEMENTE ENLAZADA

- Es una colección de nodos en la cual cada nodo tiene dos punteros , uno de ellos apuntando a su predecesor (li) y otro a su sucesor (ld). Por medio de estos punteros se podrá avanzar o retroceder a través de lista , según se tomen las direcciones de uno u otro puntero.



a) Esta es una demostración de un nodo con dos apuntadores apuntando a su predecesor (ligaiqz) y otro a su sucesor (ligader).

b) Es una lista doblemente enlazada que almacena apellidos, donde utilizamos dos apuntadores P y F que apuntan al principio y al final.





IMPORTANCIA DE LAS LISTAS DOBLEMENTE ENLAZADAS

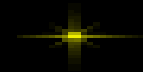
- Nos permite almacenar datos de una forma organizada .
- Es una estructura TDA dinámica.
- Cada nodo de la lista doblemente enlazada contiene dos punteros, de forma que uno apunta al siguiente nodo y el otro predecesor “permite que se pueda recorrer la lista en ambos sentidos”.

TIPOS DE LISTAS DOBLEMENTE ENLAZADAS:

***Listas dobles lineales:** en este tipo de lista doble, tanto el puntero izquierdo del primer nodo como el derecho del ultimo nodo apuntan a NIL o NULL.

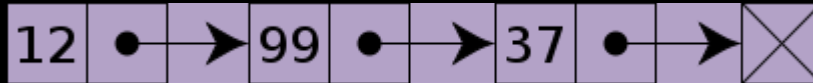
***Listas dobles circulares:** en este tipo de lista doble, el puntero izquierdo del primer nodo apunta al ultimo nodo de la lista y el puntero derecho del ultimo nodo apunta al primer nodo de la lista.

A continuación se muestran ejemplos:



LISTA ENLAZADA SIMPLE

La lista enlazada básica es la **lista enlazada simple** la cual tiene un enlace por nodo. Este enlace apunta al siguiente nodo en la lista, o al valor NULL o a la lista vacía, si es el último nodo.



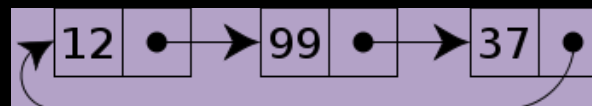
LISTAS ENLAZADAS DOBLES

Cada nodo tiene dos enlaces: uno apunta al nodo anterior, o apunta al valor NULL o a la lista vacía si es el primer nodo; y otro que apunta al siguiente nodo siguiente, o apunta al valor NULL o a la lista vacía si es el último nodo.




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 un lista enlazada circular podemos empezar por cualquier nodo y seguir la lista en cualquier dirección hasta que se regrese hasta el nodo original. Desde otro punto de vista, las listas enlazadas circulares pueden ser vistas como listas sin comienzo ni fin. Este tipo de listas es el más usado para dirigir buffers para “ingerir” datos, y para visitar todos los nodos de una lista a partir de uno dado.





OPERACIONES CON LISTAS DOBLEMENTE ENLAZADAS (LIGADAS)

- *Recorrido de lista
 - *Inserción de un elemento
 - *Borrado de un elemento
 - *Búsqueda
- 

Recorrido de una lista doblemente enlazada: . Esta operación consiste en visitar cada uno de los nodos que forman la lista . Para recorrer todos los nodos de la lista, se comienza con el primero, se toma el valor del campo liga para avanzar al segundo nodo, el campo liga de este nodo nos dará la dirección del tercer nodo, y así sucesivamente.

Inserción en listas doblemente enlazadas: consiste en agregar un nuevo nodo a la lista y establecer los apuntadores correspondientes . No es valido en una lista vacía. Se puede llevar acabo :

- ☀ Al inicio de la lista doblemente enlazada.
- ☀ Al final de la lista doblemente enlazada.
- ☀ Antes / después de un nodo con información X.

Eliminación en listas doblemente enlazadas: . La operación de borrado consiste en quitar un nodo de la lista, redefiniendo las ligas que correspondan. Se pueden presentar cuatro casos:

Eliminar el primer nodo.

Eliminar el último nodo.

Eliminar un nodo con cierta información.

Eliminar el nodo anterior o posterior al nodo cierta con información.

Búsqueda: Esta operación consiste en visitar cada uno de los nodos, tomando al campo liga como puntero al siguiente nodo a visitar.

Operaciones sobre una lista doblemente ligada en representación ligada

```
clase ListaDobleLigada
{
    Nodo cabeza
    ListaDobleLigada ()
    Nodo busca (Objeto x, bool éxito)
    void inserta (Objeto x)

    void borra (Objeto x)
}
```

Crear:

El procedimiento para crear una lista doblemente ligada es similar a la función constructora de una lista ligada:

```
ListaDobleLigada ()
{
    cabeza ← nil
}
```

Vacía:

De forma semejante el método para determinar si una lista doblemente ligada está o no vacía es:

```
bool vacía ()
//regresa verdadero si la lista se encuentra vacía, en otro caso regresa falso
comienza
vacía ← cabeza = nil
termina
```

Buscar: El algoritmo para buscar un elemento en una lista doblemente ligada es semejante al algoritmo de búsqueda en una lista ligada, pero aprovechando las ventajas que se tienen, cuando el elemento que se busca no se encuentra se regresará la posición del elemento que debería ir antes de él.

Para desarrollar este algoritmo considere que la lista doblemente ligada está ordenada en forma creciente y que no existen elementos repetidos.

Nodo busca (Objeto x, var bool éxito)

//busca un elemento x en la lista; en caso de que exista, regresa una //referencia al elemento que lo contiene; en caso contrario, regresa un

//referencia al elemento que debería ir antes que él
comienza

si cabeza = nil entonces // la lista es vacía

comienza

aux ← nil

éxito ← falso

termina

otro

comienza

aux ← cabeza

mientras **aux. ligader** <> nil & aux.info < x

aux ← aux. ligader //se avanza un elemento

éxito ← aux.info = x

si aux.info > x entonces

aux ← aux. ligaizq

termina

regresa aux

termina



Insertar:

El algoritmo para insertar que será desarrollado no permitirá elementos repetidos y conservará un orden lineal en sus elementos.

El algoritmo para insertar un nodo en una lista doblemente ligada será dividido en los siguientes casos diferentes:

- a) La lista en la cual se desea insertar el elemento se encuentra vacía
- b) El elemento que se desea insertar deberá ser la cabeza de la lista
- c) El elemento que se desea insertar deberá ser la cola de la lista
- d) El elemento que se desea insertar deberá ocupar una posición intermedia

En cada caso, se supondrá que el elemento no se encuentra y que esto fue determinado por el método busca, anteriormente desarrollado, que regresa la posición del nodo que debería ir a la izquierda (llamado ant) del que se desea insertar, esto es:

ant ---- busca (x, éxito)

si éxito entonces

error (el elemento se encuentra)

otro

a or b or c or d

donde a, b, c y d son mutuamente excluyentes.

a) La lista en la que se desea insertar se encuentra vacía

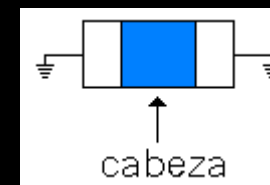
En este caso se debe:

0. Determinar que la lista se encuentra vacía

1. Crear un nodo con la información que se desea insertar y cuyas ligas referencien a nil

2. Referenciar la cabeza a este nuevo nodo

Después de insertar el nodo, gráficamente la lista deberá ser:



Borrar:

El algoritmo debe considerar que cuando se desea eliminar un elemento de una lista doblemente ligada este puede ser:

- a) el único elemento
- b) la cabeza de la lista
- c) la cola de la lista
- d) un nodo intermedio

Para cada uno de estos casos puede suponerse que el elemento forma parte de la lista; lo cual fue determinado a través del método busca de la siguiente forma:

nodo ---- busca (x, éxito)

si -éxito entonces

error (el elemento no se encuentra)

otro

a or b or c or d

EJEMPLO

Insertar un nuevo dato en una posición determinada.

/* obtiene el numero de nodos de la lista */

length = g_list_length (list);

g_print ("\nEscribe el numero de indice donde se insertara el dato (el indice máximo es %d): ", length); scanf ("%d", &index);

/* inserta el valor en la posición indicada */

if (index < length) {

list = g_list_insert (list, GINT_TO_POINTER (valué), index);

print_list (list); }

EJERCICIO:

HACER UNA LISTA LIGADA ISERTANDO ELEMENTOS EN ORDEN CRECIENTE

FIN

