Universidad Centroamericana “José Simeón Cañas”

Análisis de Algoritmos Ciclo 02/2019

Semana 3 - 26 al 30 de agosto del 2019

Guia de laboratorio 2

Repaso:

Listas

Pilas

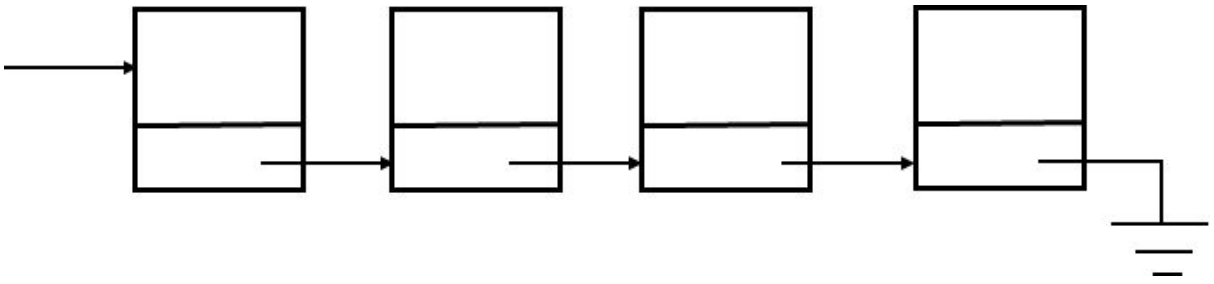
Colas

Las estructuras de datos como los arreglos se denominan **estáticas**. Reciben este nombre debido a que durante la compilación se les asigna un espacio de memoria, y éste permanece inalterable durante la ejecución del programa.

La principal ventaja de manejar un tipo **dinámico** es que se pueden adquirir posiciones de memoria a medida que se necesitan; estas se liberan cuando ya no se requieren.

Listas Simplemente enlazadas

Es una estructura simple que permite guardar un número indefinido de datos y estos datos están enlazados por punteros en este caso al ser lista de tipo simple se tendrá que sólo podrá ser recorrida en una dirección.



Cada elemento es llamado Nodo y toda lista tiene siempre una cabeza y cola siendo la primera posición en la lista y la última respectivamente.

Para los ejemplos de esta práctica, trabajaremos con la siguiente estructura:

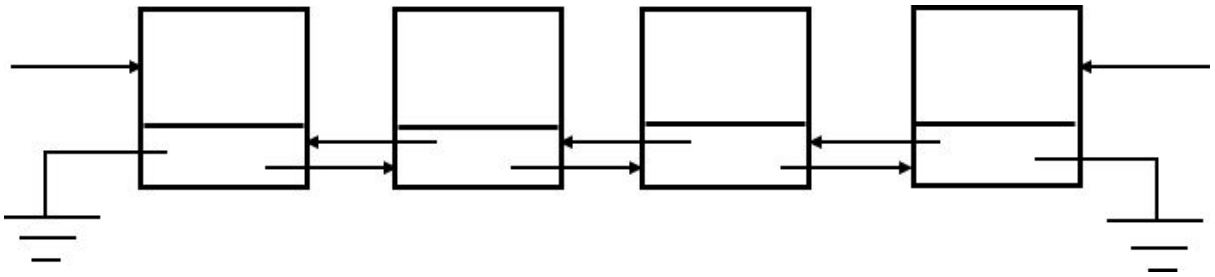
```
struct Nodo{
    int dato;
    Nodo *sig;
}*inicio;
```

Clase controladora de la lista con sus métodos triviales:

```
class ListaEnlazada{

    public:
        Nodo* crearNodo(int valor){
            Nodo *n = new Nodo;
            n->dato = valor;
            n->sig = nullptr;
            return n;
        }

        void mostrarLista(){
            Nodo *temp = inicio;
            if(!inicio)
                cout<<"La lista no posee elementos"<<endl;
            else
                while(temp){
                    cout<<temp->dato<<", ";
                    temp = temp ->sig;
                }
        }

        void agregarElemento(int valor){
            Nodo *n = crearNodo(valor), *temp;
            if(!inicio)
                inicio = n;
            else{
                for(temp=inicio; temp->sig; temp = temp->sig);
                temp->sig = n;
            }
        }

        ListaEnlazada(){
            inicio=nullptr;
        }

};
```

Listas Doblemente enlazadas

Las listas dobles siguen el mismo concepto que las simples pero el cambio que tienen es en la estructura básica para manejarlas ya que se tienen dos punteros por variable uno al siguiente y otro al anterior lo que permite recorrer la lista del inicio al final o vice versa.



La estructura que utilizamos tiene que ser modificada:

```
struct Nodo{
    int dato;
    Nodo *sig;
    Nodo *prev;
}*inicio, *fin;
```

Clase controladora de la lista con sus métodos triviales:

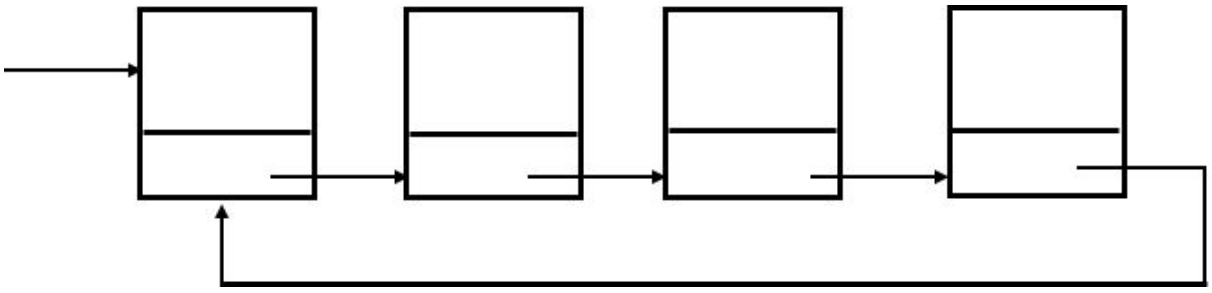
```
            void agregarElemento(int valor){
                Nodo *n = crearNodo(valor);
                if(!inicio){
                    inicio = n;
                    fin = n;
                }
                else{
                    fin->sig=n;
                    n->prev=fin;
                    fin=n;
                }
            }

Nodo* crearNodo(int valor){
    Nodo *n = new Nodo;
    n->dato = valor;
    n->sig = nullptr;
    n->prev = nullptr;
    return n;
}
void mostrarListaAdelante(){
    Nodo *temp = inicio;
    if(!inicio)
        cout<<"La lista no posee elementos"<<endl;
    else
        while(temp){
            cout<<temp->dato<<", ";
            temp = temp ->sig;
        }
}
void mostrarListaAtras(){
    Nodo *temp = fin;
    if(!inicio)
        cout<<"La lista no posee elementos"<<endl;
    else
        while(temp){
            cout<<temp->dato<<", ";
            temp = temp ->prev;
        }
}
```

Ejemplo:

Implemente una función que dada una lista doblemente enlazada L1, y un valor X, eliminar el nodo anterior al nodo con el valor X.

Listas circulares

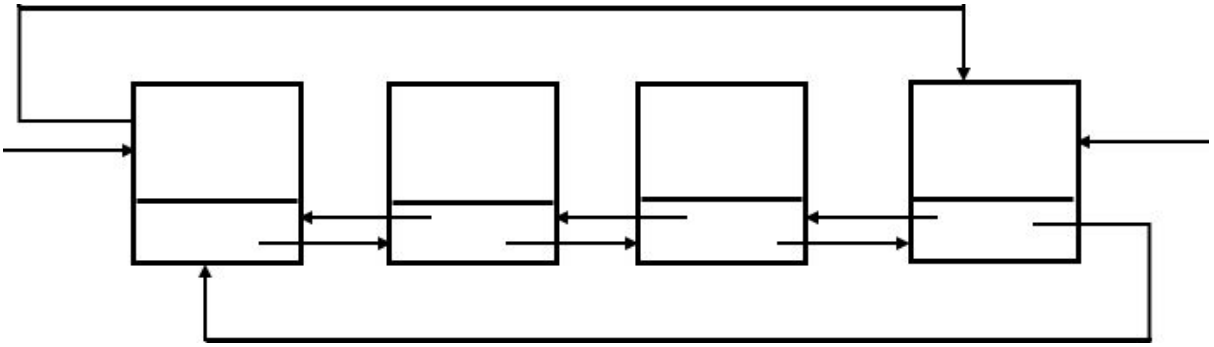
El último elemento de la lista apunta al primero, en lugar de apuntar al vacío.



Es importante señalar que al momento de recorrer este tipo de listas, se necesita considerar algún criterio para detectar cuándo se han visitado todos los nodos de la lista, con el fin de evitar caer en bucles infinitos.

```
Nodo* crearNodo(int valor){
    Nodo *n = new Nodo;
    n->dato=valor;
    n->sig=nullptr;
    return n;
}
void mostrarLista(){
    Nodo *temp = fin->sig;
    if(!fin)
        cout<<"La lista no posee elementos"<<endl;
    while(temp!=fin){
        cout<<temp->dato<<", ";
        temp = temp->sig;
    }
    cout<<temp->dato;
}
void agregarElemento(int valor){
    Nodo *n = crearNodo(valor);
    if(!fin){
        fin = n;
        n->sig = fin;
    }else{
        n->sig=fin->sig;
        fin->sig = n;
        fin=n;
    }
}
```

Listas circulares doblemente enlazadas



```
Nodo* crearNodo(int valor){
    Nodo *n = new Nodo;
    n->dato = valor;
    n->sig = nullptr;
    n->prev = nullptr;
    return n;
}
void mostrarListaAdelante(){
    Nodo *temp = inicio;
    if((!inicio) && (inicio==fin))
        cout<<"La lista no posee elementos"<<endl;
    else {
        while (temp->sig != inicio) {
            cout << temp->dato << ", ";
            temp = temp->sig;
        }
        cout << temp->dato;
    }
}
```

Ejercicio:

Se tiene una lista, el campo dato es un registro (estructura) con los campos de un alumno: nombre, edad, sexo. Escribir una función para transformar la lista de tal forma que si el primer nodo es de un alumno de sexo masculino el siguiente sea de sexo femenino.

Pilas

Estructura lineal de datos en la que los elementos se eliminan en orden inverso al que se insertaron.

```
class Pila{
    public:
        Nodo* crearNodo(int valor){
            Nodo *n = new Nodo;
            n->dato = valor;
            n->sig = nullptr;
            return n;
        }
        void push(int valor){
            Nodo *n = crearNodo(valor);
            n->sig = inicio;
            inicio=n;
        }
        void pop(){
            Nodo *temp = inicio;
            inicio = inicio->sig;
            free(temp);
        }
        void mostrarPila(){
            Nodo *temp = inicio;
            if(!inicio)
                cout<<"Pila vacia"<<endl;
            else
                while(temp){
                    cout<<temp->dato<<", ";
                    temp = temp ->sig;
                }
        }
        Pila(){
            inicio= nullptr;
        }
};
```

Colas

Estructura lineal de datos en la que los elementos se eliminan en el mismo orden al que se insertaron.

```
Nodo* crearNodo(int valor){
    Nodo *n = new Nodo;
    n->dato = valor;
    n->sig = nullptr;
    return n;
}
void push(int valor){
    Nodo *n = crearNodo(valor);
    if(!inicio){
        inicio=n;
        fin=n;
    }else{
        fin->sig=n;
        fin=n;
    }
}
void pop(){
    Nodo *temp = inicio;
    inicio = inicio->sig;
    free(temp);
}
void mostrarCola(){
    Nodo *temp = inicio;
    if(!inicio)
        cout<<"Cola vacia"<<endl;
    else
        while(temp){
            cout<<temp->dato<<", ";
            temp = temp ->sig;
        }
}
```

Tarea

Realizar la implementación en C++ del problema que le indicará su instructor y subir la entrega a la plataforma.