Implementaciones

Definición del Nodo

```
struct Nodo
{
   int info;
   Nodo* sgte;
};
```

Pilas

Push

```
void push Nodo*& p, int v){
      Nodo*q = new Nodo();
      q->info = v;
      q->sgte= p;
      p = q;
      return;
}
```

Pop

```
int pop(Nodo*& p){
      int v;
      Nodo*q = p;
      v = q->info;
      p = q->sgte;
      delete q;
      return v;
}
```

Colas

agregar

```
void agregar(Nodo* &fte, Nodo* & fin, int v){
	Nodo* p = new Nodo();
	p->info = v;
	p->sgte = NULL;
	if (fte == NULL)
		fte = p;
	else
		fin->sgte = p;
	fin = p;
	return;
}
```

Suprimir

```
int suprimir(Nodo*& fte, Nodo*& fin){
	int v;
	Nodo*q = fte;
	v = q->info;
	fte = q->sgte;
	if(fte == NULL) fin = NULL; //si queda vacia el fin
apunta a null
	delete q;
	return v;
}
```

```
	fte = q->sgte;
	if(fte == NULL) fin = NULL; //si queda vacia el fin
apunta a null
	delete q;
	return v;}
```

Listas ordenadas simplemente enlazadas

Insertar el primer nodo

```
Nodo* insertaPrimero(Nodo*& l, int x){
   Nodo* p = new Nodo();
   p->info = x;
   p->sig = NULL;

   l = p;
   return p;
}
```

La función anterior es un caso particular de la siguiente, porque?

Insertar delante del primer nodo

```
Nodo* InsertaDelante(Nodo*& l, int x){
   Nodo* p = new Nodo();
   p->info = x;
   p->sgte = l;

   l = p;
   return p;
}
```

Insertar alfinal (supone la lista no vacia)

```
Nodo* InsertaAlFinal(Nodo*& l, int x){
      Nodo* nuevo = new Nodo();
      Nodo*p= l;
      nuevo->info = x;


      while(p->sig!=NULL ){
         p = aux->sig;
      }
      nuevo->sgte = NULL;
      p->sgte = nuevo;
      return nuevo;
      }
```

Insertar en medio de dos conservando el orden creciente (supone la lista no vacia)

```
Nodo* InsertaEnMedioDeDos(Nodo*& l, int x){
      Nodo* nuevo = new Nodo();
      Nodo*p= l;
      nuevo->info = x;
      while(x>p->sgte->info){
         p = aux->sgte;
      }
      nuevo->sgte = p->sgte;
      p->sgte = nuevo;
      return nuevo;
      }
```

La función que sigue contiene a las dos anteriores, está de acuerdo? Justifique.

Insertar en medio (supone la lista no vacia)

```
Nodo* InsertaEnMedio(Nodo*& l, int x){
      Nodo* nuevo = new Nodo();
      Nodo*p= l;
      nuevo->info = x;
      while(p->sgte!=NULL && x>p->sgte->info){
         p = aux->sgte;
      }
      nuevo->sgte = p->sgte;
      p->sgte = nuevo;
      return nuevo;
}
```

Mostrar

Muestra el contenido sin perder la lista, se envía por valor

```
void mostrar(Nodo* l){
      while( aux!=NULL ){
      cout << aux->info << endl;
      aux = aux->sig;
   }
}
```

Vaciar contenido de una lista

```
void eliminar(Nodo*& l)
{
   Nodo* p;
   while( l!=NULL ){
      p = l;
      l = p-> sgte;
      delete p;
    }
}
```

Insertar un nodo con un dato simple y orden creciante

```
Nodo* insertarOrdenado(Nodo*& l, int v){
   Nodo* nuevo = NULL;
   If(l==NULL || v < l->info)
      Nuevo = insertaPrimero(l,v);
   else
      Nuevo = insertaEnMedio(l,v);
   return Nuevo;
}
```

Buscar En Lista

Busca un valor, retornando la posición, en caso de encontrarlo o NULL en caso que este no esté.

```
//pregunto por distinto a efectos de independizarme del orden
Nodo* buscar(Nodo* l, int v)
{
   Nodo* p = l;
   while( p!=NULL && p->info!=v )
      p = p->sgte;

   return aux;
}
```

Insertar sin repetir la clave

```
Nodo* buscaOInserta(Nodo*& l, int v){
   Nodo* x = buscar(l,v);//lo busca…

   if( x ) x = insertarOrdenado(p,v);//si no lo encuentra lo
inserta

   return x;
}
```

Esta función es de importancia en particula en dos patrones algorítmicos particulares, cargar sin repetir para acumular y lista de listas, una estructura compleja que en el campo de la información tiene un puntero a una lista auxiliar

Eliminar de Nodo

Elimina de la lista el valor x, en caso de encontrarlo, retorna verdadero en esta caso y falso en caso contrario

```
bool eliminar(Nodo*& l, int x){
   Nodo* p = l;
   Nodo* q = NULL;
   bool enc = false;
   while( p!=NULL && p->info!= x ){
      q = p;
      p = p->sgte;
   }

   if( p!=NULL ){//en este caso esta en P y hay que
eliminarlo
      enc = true;
      if(p==l)//pero hay que verificar si no esta en la
primera posicion
       l = p->sgte;
      else
       q->sgte = p->sgte;
      delete p;
 return enc;
}
```

Plantillas

Este tema NO se evalúa, se marca en rojo y subrayado lo que se modifica respecto del que no utiliza plantillas
El nodo

```
template <typename T> struct Nodo
{
   T info;
   Nodo<T>* sig;
};
```

Listas ordenadas simplemente enlazadas

Insertar delante del primer nodo

```
template<typenam<T> Nodo<T>* InsertaDelante(Nodo<T>*& l, T
x){
   Nodo<T>* p = new Nodo();
   p->info = x;
   p->sgte = l;
   l = p;
   return p;
}
```

Insertar alfinal (supone la lista no vacia)

```
template<typenam<T> Nodo* InsertaAlFinal(Nodo<T>*& l, T x){
      Nodo<T>* nuevo = new Nodo<T>();
      Nodo<T>*p= l;
      nuevo->info = x;
```

```
        while(p->sig!=NULL ){
           p = aux->sig;
        }
        nuevo->sgte = NULL;
        p->sgte = nuevo;
        return nuevo;
        }
```

```
template <typename T> void push(Nodo<T>*& p, T v){

        Nodo<T>* q = new Nodo<T>();
        q->info = v;
        q->sgte= p;
        p = q;
        return;
}
```

```
template <typename T> T push(Nodo<T>*& p){

        T v;
        Nodo<T>*q = p;
        v = q->info;
        p = q->sgte;
        delete q;
        return v;
}
```

```
Ejemplo de uso
Nodo* p = new Nodo()// invocación a la función sin plantilla
Nodo<int>* q = new Nodo<int>(); invoca a la funcion con
plantilla con int
Nodo<float>* r = new Nodo<float>(); invoca a funcion con
plantilla con float
int a = pop(p);
int b = pop<int>(q);
float c = pop<float>(R);
```

Esta función agrega un nuevo concepto, con que criterio se ordena. Se debe determinar si es creciente o decreciente, si es un dato simple o un campo de una struct, esto se resuelve con un nuevo parámetro,un puntero a una función.

Insertar en medio (supone la lista no vacia)

```
template<typenam<T> Nodo* InsertaEnMedio(Nodo*& l, T x, int (*criterio)(T,T)){
      Nodo<T>* nuevo = new Nodo<T>();
      Nodo<T>*p= l;
      nuevo->info = x;
      while(p->sgte!=NULL && criterio(v,p->sgte->info)>0){
         p = p->sgte;
      }
      nuevo->sgte = p->sgte;
      p->sgte = nuevo;
      return nuevo;
}
```

```
Ejemplo de uso
Struct tr{ int C1; float C2};

Int CriterioDatoSimpleCrec(int a, int b{
 return a-b;//si el valor a colocar es mayor a lo del nodo sgte sigue
}
Int CriterioDatoSimpleDecreciente(int a, int b{
 return b-a;//al reves
}
Int CriterioC1Creciente(tr a, tr b{
 return a.C1-b.C1;es por un campo determinado y creciente
}

Nodo<int>* p=InsertarEnMedio(L1,4,CriterioDatoSimpleCrec)
En forma similar para todos los demás.
```