

Listas Enlazadas en C++:

Paso	Ejemplo	
	Tipos Básicos	Estructuras
Definición del Tipo	<pre>typedef int *TPInt; typedef char *TPChar;</pre>	<pre>struct Tcomplex { double p_r; double p_i; }; typedef Tcomplex *TPComplex; ó typedef struct Tcomplex *TPComplex; struct Tcomplex { double p_r; double p_i; };</pre>
Declaración de variables	<pre>TPInt p_int; TPChar p_char;</pre>	<pre>TPComplex p_complex;</pre>
Petición de memoria (y comprobación de que la ha asignado)	<pre>p_int = new(int); p_char = new(char); if((p_int == NULL) (p_char == NULL)) { cout << "No memoria"; }</pre>	<pre>p_complex = new(Tcomplex); if(p_complex == NULL) { cout << "No hay memoria" << endl; }</pre>
Acceso I: Modificación	<pre>*p_int = 7; cin >> *p_char;</pre>	<pre>p_complex->p_r = 4.6;</pre>
Acceso II: Consulta	<pre>cout << *p_int; un_char = *p_char;</pre>	<pre>parte_img = p_complex->p_i;</pre>
Borrado	<pre>delete(p_int); delete(p_char);</pre>	<pre>delete(p_complex);</pre>

Declaración del Tipo:

```
typedef ... Tdatos;
```

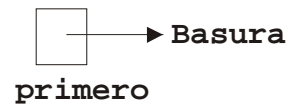
```
struct Tnodo
{
    Tdatos datos;
    Tnodo *sig;
};
typedef Tnodo *Tlista;
```

```
typedef ... Tdatos;
```

```
typedef struct Tnodo *Tlista;
struct Tnodo
{
    Tdatos datos;
    Tnodo *sig;
};
```

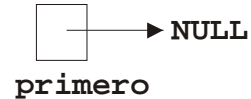
Definición de la Variable:

```
Tlista primero;
```



Inicialización de la Lista:

```
Tlista crear()
{
    return NULL;
}
...
primero = crear();
```

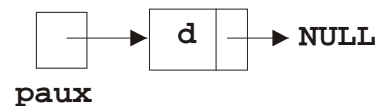


Insertar Ordenado:

1. Crear Nodo a insertar

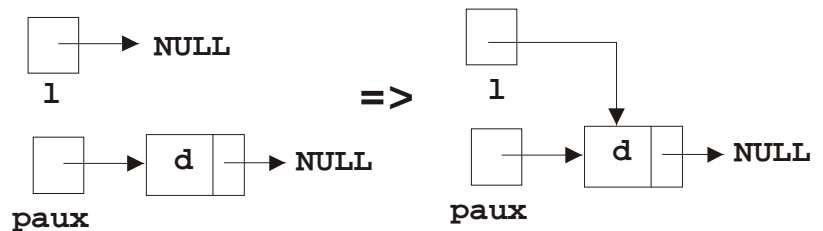
```
void insertar(Tlista &l, Tdato d)
{
    Tlista paux;
    paux = new(Tnodo);

    if(paux == NULL)
    {
        cout << "ERROR";
    }
    else
    {
        paux->dato = d;
        paux->sig = NULL;
    }
}
```



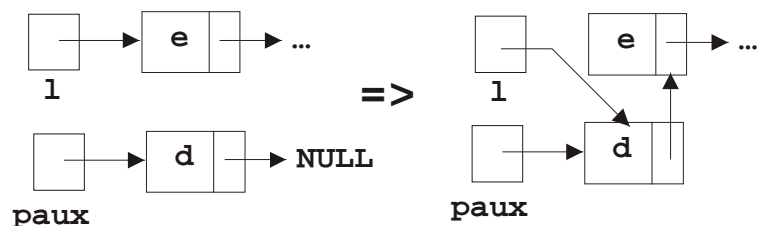
2. Si la lista es vacía introducimos al principio:

```
if( l == NULL )
{
    l = paux;
}
```



3. Si no es vacía comprobamos si hay que añadir al principio:

```
if( l.dato > d )
{
    paux->sig = l;
    l = paux;
}
```



4. Insertamos en medio

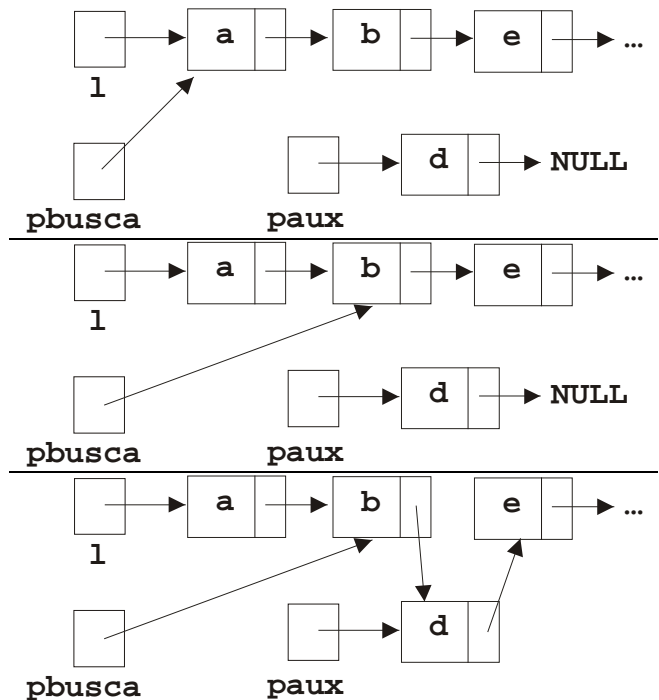
```

/* Creamos un puntero auxiliar
para buscar la posición donde
insertar */
pbusca = l;

/* Recorremos la lista buscando
donde insertar */
while( (pbusca->sig != NULL) &&
       (pbusca->sig->datos < d))
{
    pbusca = pbusca->sig;
}

/* Una vez encontrada la posición
insertamos el nuevo nodo */
paux->sig = pbusca->sig;
pbusca->sig = paux;

```



Borrar:

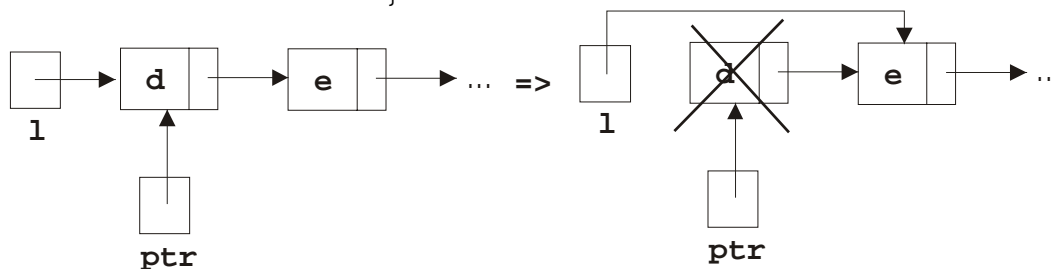
Borrar es similar a la insertar. Primero se comprueba si es vacía, en tal caso no hacemos nada. Si no está vacía buscamos el elemento que queremos borrar, y una vez localizado, lo sacamos de la lista y liberamos la memoria que tenía asignada-

1. Borrado del primer elemento

```

if(l->dato == d)
{
    ptr = l;
    l = l->sig;
    delete(ptr);
}

```



2. Borrado de un elemento intermedio

```

/* Creamos dos punteros
auxiliares uno que examine la
posición actual y otro que
indique la posición anterior
*/

```

```

pact = l->sig;
pant = l;

```

```

/* Recorremos la lista
buscando el nodo a borrar*/
while((pact != NULL) &&
      (pact->dato != d))

```

```

{
    pant = pact;
    pact = pact->sig;
}

```

```

/* Una vez encontrada la
posición borramos el nodo */
if(pact != NULL)

```

```

{
    pant->sig = pact->sig;
    delete(pact);
}

```

