

Punteros

Listas Enlazadas

Tomás Peiretti

Listas enlazadas

Una lista enlazada es una **estructura de datos dinámica** que almacena los datos de forma secuencial, no contigua.

- Que sea dinámica significa que su tamaño físico puede variar durante tiempo de ejecución.

Operación	Arreglos	Listas enlazadas
Buscar un elemento	$O(N)$	$O(N)$
Acceder a un elemento	$O(1)$	$O(N)$
Eliminar un elemento	$O(N)$	$O(1)$
Agregar un elemento	$O(N)$	$O(1)$



Listas enlazadas: implementación



```
1 // cada elemento (nodo) contiene datos/informacion del elemento y
2 // la referencia (un puntero) al nodo siguiente.
3 // Ejemplo: Lista enlazada de enteros
4 struct Nodo {
5     int num;
6     Nodo * sig;
7 };
8 // Ejemplo: Lista enlazada de Personas
9 struct Persona {
10     string nombre;
11     int edad;
12     int altura;
13 };
14
15 struct NodoP {
16     Persona p;
17     NodoP * sig;
18 };
19
20 int main() {
21     // para manejar la lista enlazada,
22     // utilizaremos un puntero cabecera (HEAD)
23     // el cual apuntara al primer elemento de la lista
24     Nodo * listaEnteros;
25     NodoP * listaPersonas;
26 }
```

Listas enlazadas: creación y eliminación de nodos

Para crear un nuevo nodo/elemento, debemos reservar memoria. Para esto, utilizaremos la instrucción **new**.

new permite reservar memoria para una cantidad específica de bytes y retorna un puntero al objeto creado. En el caso de que no haya memoria, retorna un puntero NULL.

```
1 struct Nodo {  
2     int num;  
3     Nodo * sig;  
4 };  
5  
6 int main() {  
7     // creamos una lista enlazada con un unico elemento  
8     Nodo * lista = new Nodo;  
9     // chequeamos si se pudo crear el nodo  
10    if (lista != NULL) {  
11        lista->num = 100;  
12        lista->sig = NULL;  
13    } else {  
14        cout << "ERROR - NO HAY MEMORIA DISPONIBLE" << endl;  
15    }  
16    // Luego, al eliminar un elemento, debemos liberar la memoria  
17    delete lista;  
18    lista = NULL;  
19 }
```

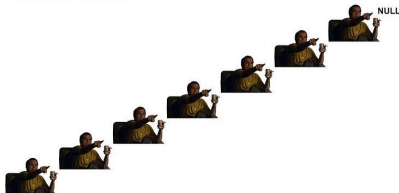
Listas enlazadas: Operaciones

¿Qué operaciones debemos saber realizar con las listas enlazadas para aprobar AEDD?

- Eliminar/agregar un elemento
- Buscar un elemento
- Unir dos listas
- Invertir una lista
- Ordenar (SelectionSort, InsertionSort, etc)

Nobody:

Linked Lists:



Operaciones: Creación de nodo

```
1 struct Nodo {
2     int num;
3     Nodo * sig = NULL;
4 };
5
6 bool agregarNodo(int num, Nodo * &lista) {
7     Nodo * nuevo = new Nodo;
8     if (nuevo == NULL) {
9         return false;
10    }
11
12    nuevo->num = num;
13    // si la lista esta vacia
14    if (lista == NULL) {
15        lista = nuevo;
16        return true;
17    }
18
19    // para agregar un nodo, debemos ir hasta el ultimo
20    Nodo * aux = lista;
21    while (aux->sig != NULL) {
22        aux = aux->sig;
23    }
24    // teniendo el ultimo nodo, agregamos el nuevo
25    aux->sig = nuevo;
26    return true;
27 }
```

Operaciones: Creación de nodo 2

```
1 bool agregarNodo(int num, int pos, Nodo * &lista) {
2     Nodo * nuevo = new Nodo;
3     if (nuevo == NULL) {
4         return false;
5     }
6     nuevo->num = num;
7     // si se desea agregar como primer elemento:
8     if (pos == 0) {
9         nuevo->sig = lista;
10        lista = nuevo;
11        return true;
12    }
13    // para agregar un nodo en pos, debemos ir hasta la posicion previa
14    Nodo * aux = lista;
15    int i = 0;
16    while (aux != NULL && i < pos - 1) {
17        aux = aux->sig;
18        i++;
19    }
20    if (aux == NULL) {
21        delete nuevo;
22        return false;
23    }
24
25    nuevo->sig = aux->sig;
26    aux->sig = nuevo;
27    return true;
28 }
```

Operaciones: Buscar un elemento

```
1 struct Nodo {
2     int num;
3     Nodo * sig = NULL;
4 };
5
6 void buscarNum(int num, Nodo * lista) {
7     int pos = 0;
8     while(lista != NULL && lista->num != num) {
9         lista = lista->sig;
10        pos++;
11    }
12
13    if (lista == NULL) {
14        cout << "No se ha encontrado a " << num << " en la
15        lista"<<endl;
16    } else {
17        cout << num << " se encuentra en el nodo " << pos <<
18        endl;
19    }
20 }
```


Operaciones: Eliminar un elemento

```
1 struct Nodo {
2     int num;
3     Nodo * sig = NULL;
4 };
5
6 void eliminar(int num, Nodo * &lista) {
7     Nodo * aux = lista;
8     Nodo * prev = NULL;
9     while (aux != NULL && aux->num != num) {
10         prev = aux;
11         aux = aux->sig;
12     }
13
14     // nada que eliminar, no se encontro
15     if (aux == NULL) {
16         return;
17     }
18
19     // el nodo a eliminar es el primero
20     if (prev == NULL) {
21         lista = aux->sig;
22     } else {
23         prev->sig = aux->sig;
24     }
25     delete aux;
26 }
```

Operaciones: Invertir una lista

```
1 struct Nodo {  
2     int num;  
3     Nodo * sig = NULL;  
4 };  
5  
6 void invertir(Nodo * &lista) {  
7     // lista vacia, nada que invertir  
8     if (lista == NULL) {  
9         return;  
10    }  
11  
12    Nodo * prev = NULL;  
13    Nodo * actual = lista;  
14  
15    while (actual != NULL) {  
16        Nodo * sig = actual->sig;  
17        actual->sig = prev;  
18        prev = actual;  
19        actual = sig;  
20    }  
21    lista = prev;  
22 }
```

- Guía de ejercicios de práctica de la cátedra