



# ESTRUCTURAS DE DATOS

## TAD Listas Lineales

1

1



## TAD: Tipo Abstracto de Dato

### **Abstraer ([www.rae.es](http://www.rae.es)):**

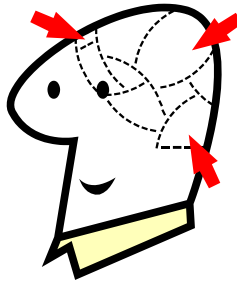
1. Separar por medio de una operación intelectual las cualidades de un objeto para considerarlas aisladamente o para considerar el mismo objeto en su pura esencia o noción.
2. Prescindir, hacer caso omiso. *Abstraer DE examinar la naturaleza de las cosas.*
3. Enajenarse de los objetos sensibles, no atender a ellos por entregarse a la consideración de lo que se tiene en el pensamiento.

2

2

## Abstracción

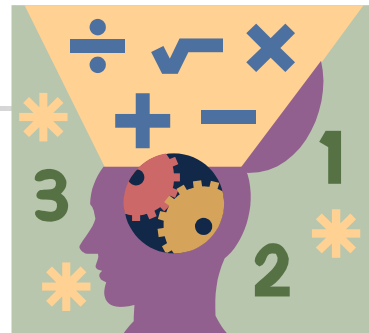
Es un mecanismo de la mente humana que permite la comprensión de fenómenos o situaciones que involucren una gran cantidad de detalles.



3

## Abstracción

Abstraer es un proceso mental sobre un objeto en estudio, compuesto de:



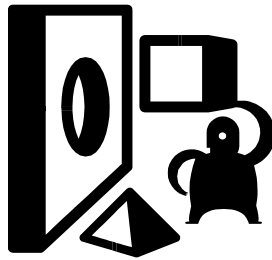
- **Destacar** los detalles relevantes
- **Ignorar** los detalles irrelevantes

4



## Tipo Abstracto de datos (TAD)

- La abstracción permite **simplificar** el análisis y resolución de un problema separando las características que son relevantes de aquellas que no lo son.



5

5



## Tipo Abstracto de datos (TAD)

- Un tipo abstracto de dato es un **modelo matemático** compuesto por una colección de **operaciones** definidas sobre un **conjunto de datos** para el modelo.
- El concepto de tipo de dato abstracto fue propuesto por John Guttag en 1974.

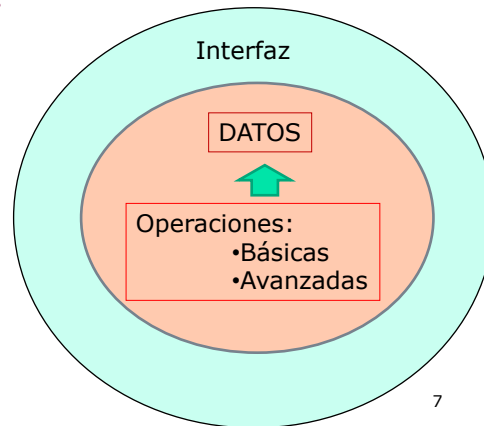
6

6



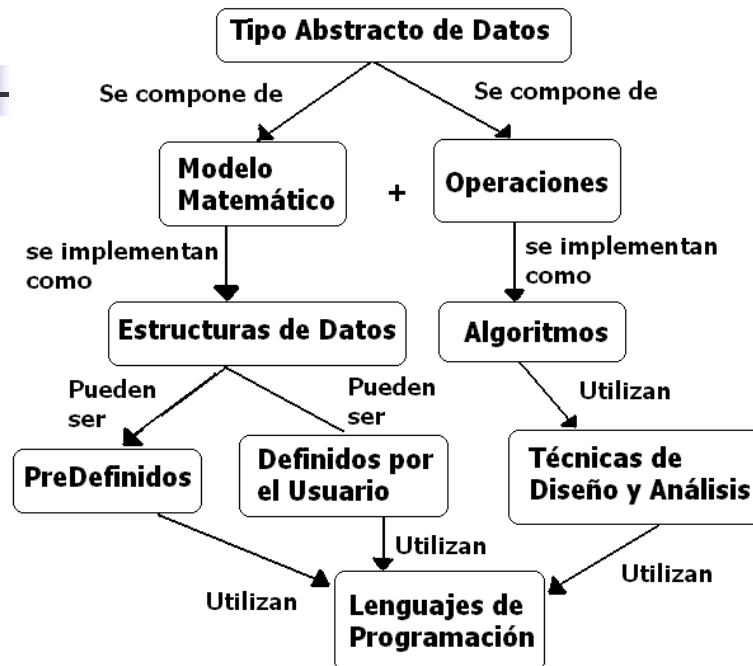
## Utilización de un TDA

- Un TDA se divide en dos partes:
  - Especificación.
  - Implementación.



7

7



8

8

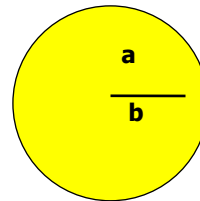


## Ejemplo: TAD para Números Racionales

¿Qué es un número racional?

•Un **número racional** es un número que puede representarse como el cociente de dos números enteros, es decir, una fracción común  **$a/b$**  con **numerador**  $a$  y **denominador** distinto de cero  $b$ .

•El término racional alude a fracción o parte de un todo.



9

9



## Ejemplo de Definición del TAD Numero Racional:

Conjunto de pares de elementos **(a,b)** de tipo entero, con  **$b \neq 0$** .

```
struct rac{  
    int a,b;  
}
```

Entonces, las representaciones serían:

$$4/5 \rightarrow (4, 5)$$

$$2 \frac{1}{3} \rightarrow (7, 3)$$

$$4 \frac{3}{4} \rightarrow (19, 4)$$

10

10



## Ejemplo:

### -Operaciones del TAD:

- CrearRacional:  $a, b = (a, b)$
- Suma:  $(a, b) + (c, d) = (a*d + b*c, b*d)$
- Resta:  $(a, b) - (c, d) = (a*d - b*c, b*d)$
- Producto:  $(a, b) * (c, d) = (a*c, b*d)$
- División:  $(a, b) / (c, d) = (a*d, b*c)$
- Numerador:  $(a, b) = a$
- Denominador:  $(a, b) = b$
- ValorReal:  $(a, b) = a/b$
- MCD:  $(a, b) \dots$
- Potencia:  $(a, b)^c = (a^c, b^c)$
- Simplifica:  $(a, b) = (a/\text{mcd}(a, b), b/\text{mcd}(a, b))$

11

11



## Recordar: Una Lista es una...

- ...Enumeración, generalmente en forma de columna, de personas, cosas, cantidades, etc., que se hace con determinado propósito.



[www.rae.es](http://www.rae.es)



12



## TDA Lista Enlazada

- Una lista es una colección de elementos ordenada de acuerdo a las posiciones de estos: *secuencia*, relación *predecesor-sucesor*.

$$\begin{array}{ccc} \text{primer elemento} & \downarrow & \\ L = \langle a_0, a_1, \dots, a_{n-1} \rangle & & \\ & \uparrow & \text{último elemento} \end{array}$$

- Por ejemplo, las listas  $\{1, 3, 7, 6\}$  y  $\{6, 3, 7, 1\}$  NO son iguales. Si fueran conjuntos, entonces sí lo serían.
- Otra diferencia con los conjuntos es que en una lista, además, pueden existir elementos repetidos.

13

## TDA Lista Enlazada

Sea  $L = (a_0, a_1, a_2, a_3, \dots, a_{n-1})$ , entonces

1.  $a_0$  es el primer elemento
2.  $a_{n-1}$  es el último elemento
3.  $a_i$  precede al elemento  $a_{i+1}$
4.  $a_i \in L$ , para  $i=0, \dots, n-1$  ( $n$  es la longitud de  $L$ )
5. Si  $n=0$  entonces se trata de la lista vacía
6. Un elemento puede insertarse o eliminarse en cualquier posición de la lista.

14



## Operaciones Básicas en Estructuras Lineales

1. **Recorrido:** Procesa c/elemento de la estructura Lista.
2. **Búsqueda:** Recupera la posición de un elemento específico dentro de la lista.
3. **Inserción:** Incorpora un nuevo elemento a la estructura Lista.
4. **Eliminación:** Elimina un elemento de la estructura Lista.
5. **Ordenación:** Ordena los elementos de la estructura con algún criterio según los valores que contiene.
6. **Mezcla:** Combina 2 ó más listas en una sola.

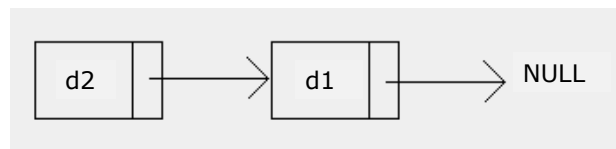
15



## Lista enlazada

Una lista enlazada es un tipo de dato auto-referenciado porque contienen un puntero o link a otro dato del mismo tipo.

Una lista puede cambiar de tamaño, su ventaja fundamental es que es flexible a la hora de reorganizar sus elementos; a cambio se ha de pagar una mayor lentitud a la hora de acceder a cualquier elemento.



16



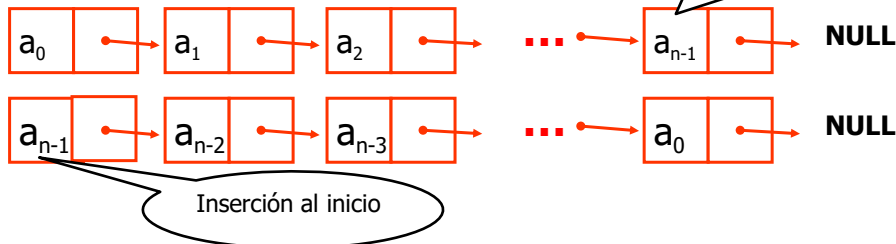
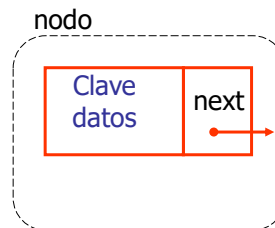
# Listas enlazadas

- Una lista enlazada se puede definir recursivamente de la siguiente manera:
  - una lista enlazada es una estructura vacía
  - un elemento de información y un enlace hacia una lista (un nodo).

17

## Lista Lineal Enlazada

- Es una estructura compuesta por una secuencia de nodos.
- Cada nodo almacena:
  - Clave
  - datos
  - link al siguiente nodo (next)
- Dos formas de implementar una lista:



18



## Listas enlazadas

- Pueden cambiar de tamaño
- Son flexibles a la hora de reorganizar sus elementos
- Son más lentas al acceder a cualquier elemento.

19



### Listas enlazadas. ***Implementación***

- Se representará una lista vacía con la constante NULL.
- Se puede definir la lista enlazada de la siguiente manera:

```
struct lista {  
    int clave;  
    struct lista *next;  
};
```

- En este caso el elemento de información es un entero.
- Se trata de una definición autorreferencial.

20



## Listas enlazadas. *Implementación*

Ejemplo: 

```
struct cl {  
    char nombre[20];  
    int edad;  
};
```

```
struct lista {  
    struct cl datos;  
    int clave;  
    struct lista *sig;  
};
```

Cuando se crea una lista debe estar vacía:

```
struct lista *L = NULL;
```

21

```
#include <stdlib.h>
```

```
struct lista {  
    int clave;  
    struct lista *sig;  
};
```

```
int main(void) {  
    struct lista *L, *p;  
    int i;  
    L = NULL;
```

```
    for (i = 4; i >= 1; i--) {  
        p = (struct lista *) malloc(sizeof(struct lista));  
        p->clave = i;  
        p->sig = L;  
        L = p;  
    }  
    return 0;  
}
```

Crea una  
lista vacía

Reserva  
memoria  
para un  
nodo

Reorganiza  
los enlaces

22



## Implementación de Listas Lineales Enlazadas

```
#include<stdio.h>
#include<stdlib.h>

typedef struct nodo
{
    int num;
    struct nodo *link;
}*NODO;
```

23

```
NODO Crear_lista(int n){ //funcion que crea una lista
    NODO aux, lista=NULL, nuevo;
    int cont=0;
    if(n>0){
        lista = new nodo;
        lista->num=rand()%100;
        lista->link = NULL;
        aux = lista;
        cont = 1;
        while (cont != n){
            nuevo = new nodo;
            nuevo->num=rand()%100;
            nuevo->link = NULL;
            aux->link = nuevo;
            aux = aux->link;
            cont++;
        }
    }
    return lista;
}
```

Creación de  
la lista  
(n elementos)

24

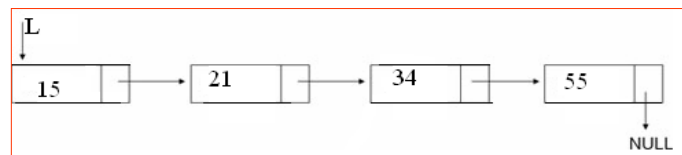
## Recorrido de una lista

- La idea es ir avanzando desde el elemento de la cabecera de lista hasta encontrar el puntero a NULL.
- Antes de acceder a la estructura lista es fundamental saber si esa estructura existe, es decir, que no está vacía.
- la lista enlazada es una estructura recursiva, y una posibilidad para su recorrido es hacerlo de forma recursiva.

25

## Ejemplo:

- L es el primer nodo o cabeza de la lista (Head).



Escriba una función que permita:

1. Retornar el valor de la suma de los elementos de la lista.
2. Imprimir los elementos de la lista en orden secuencial a partir de la cabecera de lista.

26



## Recorridos sobre listas enlazadas

```
int suma(NODO L)
{
    int aux=0;
    NODO p;
    p=L;
    while (p != NULL) {
        aux = aux + p->num;
        printf("%d",p->num);
        p=p->link;
    }
    return(aux);
}
```

27



## Recorridos sobre listas enlazadas

```
void Imprimir_lista (NODO aux)
{
    //Funcion que Imprime la lista
    if(aux != NULL)
    {
        while (aux != NULL)
        {
            printf("[%d] ", aux->num);
            aux = aux->link;
        }
        printf ("\n\n");
    }
    else printf ("Lista vacía\n\n");
}
```

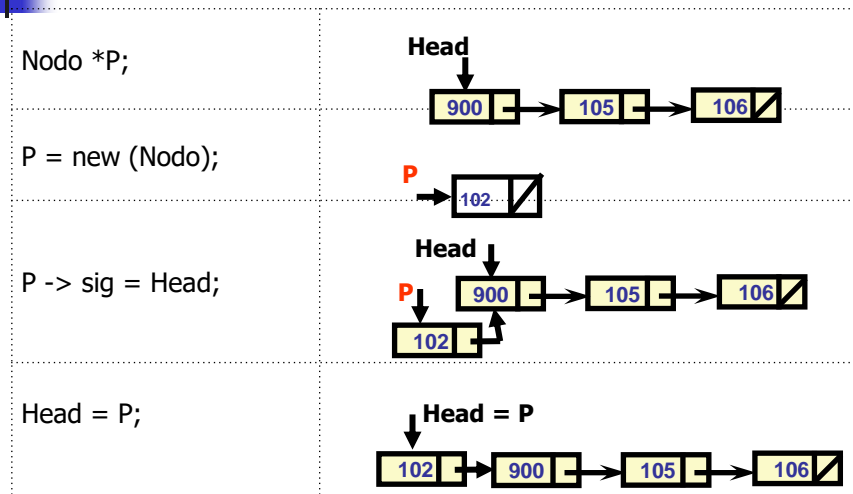
28

## Inserción

- Para insertar un nuevo elemento en la lista ordenada hay que hacerlo en el lugar que le corresponda, y esto depende del orden y de la clave escogidos. Este proceso se realiza en tres pasos:
  1. Localizar el lugar correspondiente al elemento a insertar.
  2. Reservar memoria para el nuevo nodo.
  3. Enlazarlo.

29

## Agrega un nodo al inicio de la lista

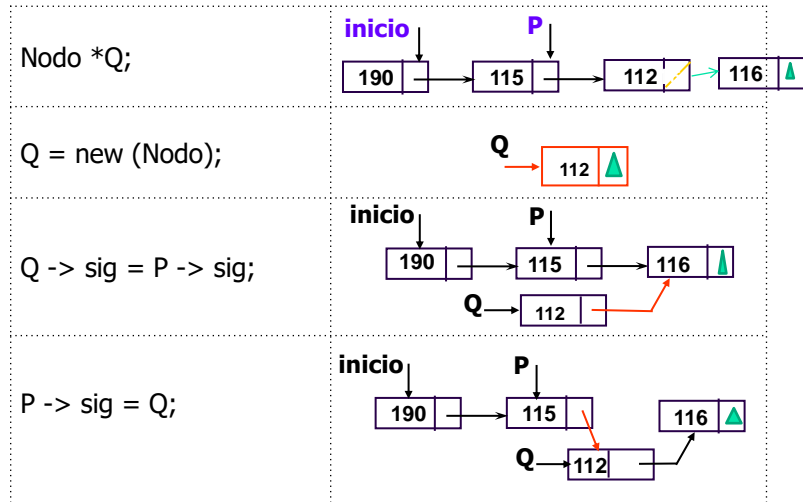


Si Head=NULL, crea el primer nodo de la lista

30



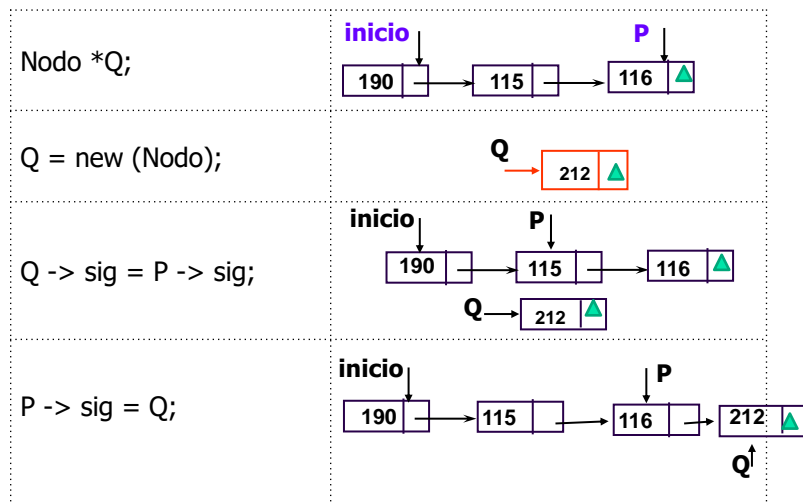
Agregar un nodo con información 112 después del nodo P



31



Agregar un nodo con información al final de la lista:



32





## Inserción al final de una lista

```
NODO insertaNodo(NODO h){
    NODO aux, nuevo;
    if (h!=NULL){
        aux=h;
        while(aux->link != NULL) aux=aux->link;
        nuevo = new nodo;
        nuevo->num=rand()%100;
        nuevo->link = NULL;
        aux->link = nuevo;
    }
    else {
        nuevo = new nodo;
        nuevo->num=rand()%100;
        nuevo->link = NULL;
        h=nuevo;
    }
    return h;
}
```

33

## Búsqueda en una lista

En el caso de trabajar con un vector, la búsqueda variaba sensiblemente según que éste estuviera o no ordenado. En el caso de una lista enlazada, la búsqueda debe hacerse mediante un recorrido de la misma, elemento a elemento, hasta o bien encontrar el elemento deseado o bien detectar el final de la lista,

```
int buscar(NODO h, int X)
{
    int flag=0;
    while(h!= NULL){
        if(h->num== X)flag=1;
        h=h->link;
    }
    return(flag);
    //si sale del while es que no está X o la lista es vacía
}
```

34



## Eliminación en una lista enlazada.

1. Búsqueda del valor X a eliminar en una lista:
2. Si está se distinguen tres casos:
  - está al inicio
  - está en una posición intermedia
  - está al final
3. Si no está, no se produce eliminación.

35



## Elimina al nodo que está después del nodo P

Nodo *Q;	
Q = P -> sig;	
P -> sig = Q -> sig;	
delete Q;	

36



## Elimina el primer nodo de la lista

Nodo *P;	
P = L;	
L = L -> sig;	
delete P;	

37



```

NODO eliminar(NODO h, int X) {
    NODO p,q;
    p=h;
    if (h!=NULL){
        if(p->num == X){
            h=h->link;
            delete(p);
        }
        else {q=p->link;
            while(q->num != X) {
                p=q;
                q=q->link;
            }
            p->link=q->link;
            delete(q);
        }
    }
    else printf ("Lista vacía\n\n");
    return(h);
}

```


38



## Para crear el menú...

```
void instrucciones( void )
{
    printf( "Introduzca su eleccion:\n"
        " 1 para crear una lista randómica de N elementos\n"
        " 2 para imprimir la lista \n"
        " 3 para insertar un nuevo elemento en la lista\n"
        " 4 para eliminar un elemento de la lista.\n"
        " 5 para invertir la lista\n"
        " 6 para terminar la ejecución\n" );
}
```

39



```
main(){
    NODO h=NULL;
    int n,s;
    char eleccion;
    instrucciones();
    scanf("%c",&eleccion); /* repite mientras el usuario no elija 6 */
    do {
        switch ( eleccion ) {
            case '1': {
                printf("Ingrese total de nodos de la lista\n");
                scanf("%d",&n);
                h=Crear_lista(n); Imprimir_lista(h);
                break;
            }
            case '2': {
                Imprimir_lista(h);
                break;
            }
            case '3': {
                h=insertaNodo(h); Imprimir_lista(h);
                break;
            }
        }
    } while (eleccion != '6');
```

Cuerpo  
principal

40



## Cuerpo principal

```
case '4': {
    printf("Ingrese el valor que desea eliminar:\n");
    scanf("%d",&n);
    s=buscar(h,n);
    if(s)h=eliminar(h,n);
    else printf("El valor no se encuentra en la lista\n");
    Imprimir_lista(h);
    break; }
case '5':{
    h=Invertir(h);
    printf("Lista invertida\n");
    Imprimir_lista (h);
    break; }
default:{
    break;
}
case '6': printf( "Fin de la ejecucion.\n" );
} /* fin de switch */
scanf("%c",&eleccion);
}
while (eleccion != '6' ); /* fin de do-while */
system("pause");
}
```