

ESTRUCTURAS DE DATOS DINÁMICAS

- Las **ESTRUCTURAS DE DATOS ESTÁTICAS** son aquellas en las que el tamaño ocupado en la memoria se define antes de que el programa se ejecute y no puede ser modificado durante la ejecución del programa.
- Las **ESTRUCTURAS DE DATOS DINÁMICAS** son aquellas en las que el tamaño podrá modificarse durante la ejecución del programa; teóricamente no hay límites a su tamaño, salvo el que impone la memoria disponible en la computadora.

Ventajas del uso de Arreglos

- Son más seguros: al no haber redimensionamientos, el riesgo de errores es menor.
- También hay menos problemas en cuanto a que datos válidos sean eliminados por error o a que existan datos no válidos en el array.
- No hay equívocos en cuanto al número de elementos que los componen y son más fáciles de seguir.

Desventajas del uso de Arreglos

- Limitan en cuanto a capacidad de maniobra frente a las circunstancias y en cuanto a posibilidades de una adaptación perfecta entre el número de datos válidos y el número de elementos del arreglo.
- No se puede modificar el tamaño del arreglo por lo que hay que considerar que Almacenar datos no válidos **es ineficiente**, al ocupar memoria y tener que realizar la gestión de elementos que no aportan nada.
- Aumenta el **riesgo de errores** de interpretación para determinar los datos existentes y datos válidos.

PUNTERO

Un puntero es una variable que contiene la dirección de memoria de otra variable.

- Un puntero es una variable que apunta o referencia a una ubicación de memoria en la cual hay datos.
- Es un tipo de dato que “apunta” a otro valor almacenado en memoria.

ESTRUCTURA DE DATOS LINEALES

Las **estructuras de datos lineales** son aquellas en las que los elementos ocupan lugares sucesivos en la estructura y cada uno de ellos tiene un único sucesor y un único predecesor, es decir, sus elementos están ubicados uno al lado del otro relacionados en forma lineal.

Hay tres tipos de **estructuras de datos lineales**:

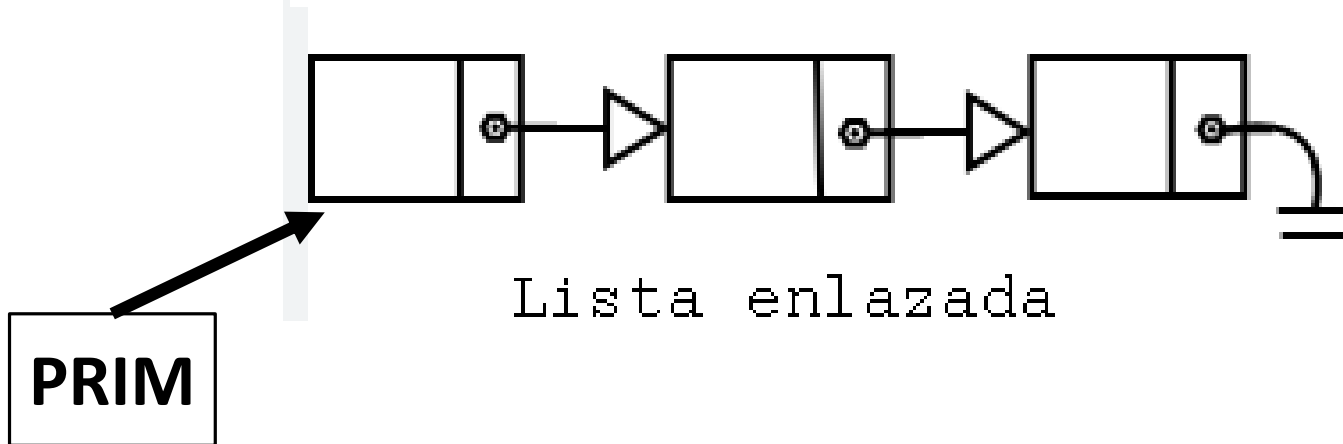
- Listas enlazadas
- Pilas
- Colas

LISTAS ENLAZADAS

Las listas enlazadas se construyen con elementos que están ubicados en una secuencia. Es decir que cada elemento se conecta con el siguiente a través de un enlace que contiene la posición del siguiente elemento. De este modo, teniendo la referencia del principio de la lista se puede acceder a todos los elementos de la misma.



Estructura de un nodo



Lista enlazada

**Acción Lista es
Ambiente**

Tipo

```
punt:puntero a elto;  
registro: elto  
    valor: entero;  
    proximo: puntero a elto;  
fin registro;
```

```
prim,p: punt;  
cant,i,dato: entero;
```


Algoritmo

i=0;

Escribir (“Ingresar cantidad de elementos de la lista”);

Leer (cant);

nuevo (p);

si (p = null) entonces escribir (“Error”);

sino

 Escribir (“Ingrese dato”);

 Leer (dato);

 *p.dato:=dato;

 *p.proximo:=null;

 prim:=p;

 i:=i+1;

 nuevo (p);

```
Mientras (p<>Null)  $\wedge$  (i<cant) hacer
    Escribir ("Ingrese dato");
    Leer (dato);
    *p.valor:=dato;
    *p.proximo:=prim;
    prim:=p;
    i:=i+1;
    Nuevo (p);
```

```
fin mientras
```

```
si (i= cant) entonces escribir ("Lista completa");
    sino escribir ("La lista tiene menos elementos");
```

```
escribir ("Mostrar los elementos de la lista");
```

```
p:=prim;
```

```
mientras (p<> Null) hacer
```

```
    escribir (*p.valor);
```

```
    p:=*p.proximo
```

```
fin mientras
```

```
sin si
```

```
fin acción
```

A partir del algoritmo anterior, indicar la cantidad de números pares e impares que tiene la lista

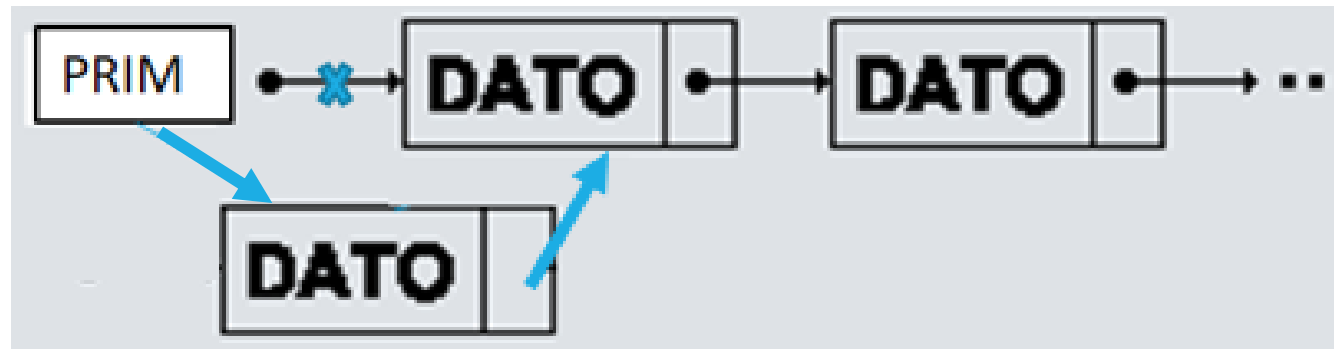
```
par:=0;
impar:=0;
p:=prim;
  mientras (p<> Null) hacer
    si (*p.valor MOD 2=0) entonces par:=par+1;
    sino impar:=impar+1;
  fin si
  p:=*p.proximo
fin mientras

escribir ("Cantidad de números pares:",par);
escribir ("Cantidad de números impares:",impar);
```

Operación Insertar al principio de la lista

A partir del algoritmo anterior realizar la operación indicada:

```
nuevo (p);  
si (p = null) entonces escribir ("Error");  
sino  
    t:=prim;  
    escribir ("Ingrese dato");  
    Leer (dato);  
    *p.valor:=dato;  
    *p.proximo:=prim;  
    prim:= p;
```



Operación Insertar al final de la lista

A partir del algoritmo anterior realizar la operación indicada:

nuevo (p);

si (p = null) entonces escribir ("Error");

sino

t:=prim;

mientras (t<> Null) hacer

aux:= t;

t:=*t.proximo;

fin mientras

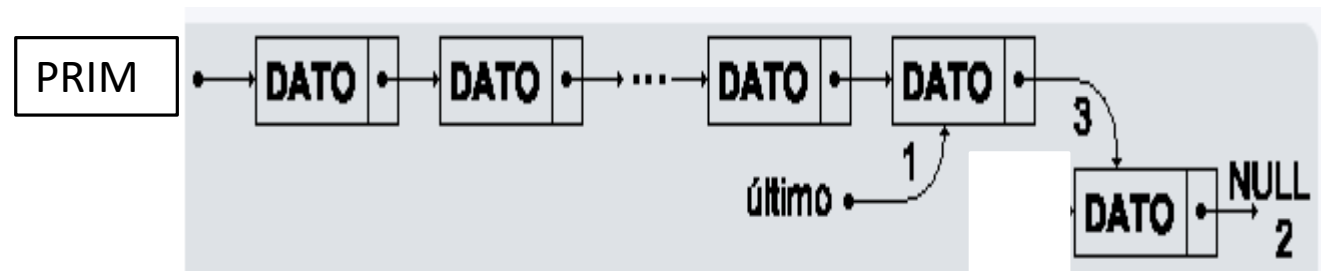
escribir ("Ingrese dato");

Leer (dato);

*p.dato:=dato;

*p.proximo:=null;

*aux.proximo:= p;



Operación Insertar al medio de la lista

A partir del algoritmo anterior realizar la operación indicada:

```
nuevo (p);  
si (p = null) entonces escribir ("Error");  
sino  
    escribir ("Ingrese dato a insertar");  
    Leer (dato);  
    t:=prim;  
  
    mientras (t<> Null )  $\wedge$  (*t.valor < dato) )hacer  
        aux:= t;  
        t:=*t.proximo;  
    fin mientras  
  
    si (t=null) entonces escribir ("No se encontró un valor menor al ingresado");  
    sino  
        *p.valor:=dato;  
        *p.proximo:=t;  
        *aux.proximo:= p;  
    fin si
```

