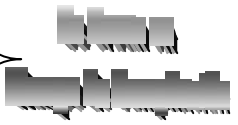
Tema 4.
Estructuras Dinámicas

Diseño de Algoritmos.

E.U. Politécnica
Departamento Lenguajes y Ciencias de la Computación.
Universidad de Málaga
José Luis Leiva Olivencia.
Despacho: I-326D (Edificio E.U.P)/ 3.2.41 (Teatinos-E.T.S.I.I.)

Introducción a las Estructuras de Datos Dinámicas

- Variables estáticas:
  - Se conoce su nombre.
  - Se conoce cuando empieza/acaba su existencia.
  - Se conoce el espacio que ocupan en memoria.



Introducción a las Estructuras de Datos Dinámicas

Problema

¿Que sucede si a priori no conocemos la cantidad de espacio de almacenamiento que vamos a precisar?

Solución ⇔ Hacer una previsión??

Ejemplo:

Tipos

```
Struct persona {char nombre[30];
                int edad;}
```

Variables

```
Struct persona poblacion[30];
```


Introducción a las Estructuras de Datos Dinámicas

- Variables anónimas:
  - No se conoce su nombre.
  - No se conoce el momento en que empieza/acaba su existencia.
  - El espacio que van a ocupar en memoria es variable.

PUNTEROS

¿Que Es un Puntero?

Variable Estática

Un puntero es una variable que almacena una dirección de memoria

Declaración de Punteros

```
Tipos
  typedef int *ptrint;
/*puntero a enteros*/
Variables
  ptrint p;
/*puntero a enteros*/
```

1ª FORMA

```
Variables
  int *p;
/*puntero a enteros*/
```

2ª FORMA

Declaración de Punteros

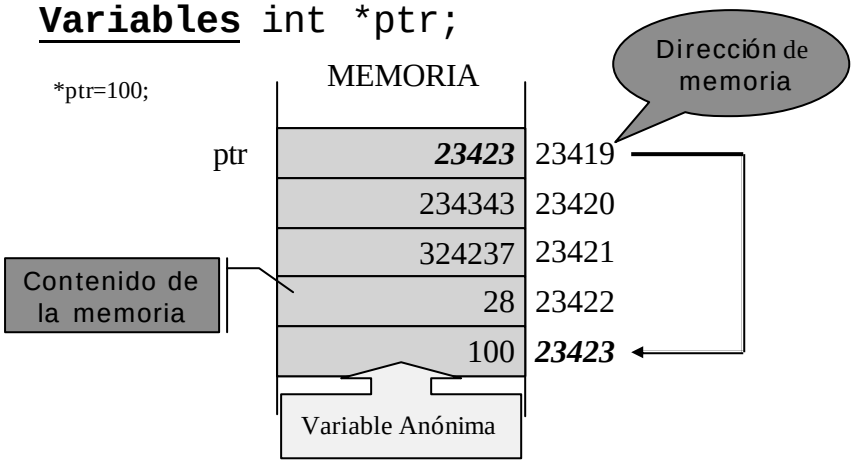
```
Typedef struct
{ int num;
  char car;
} tiporegistro;
Typedef tiporegistro *tipopuntero;
Tipopuntero p;
```

Así:
p es la dirección de un registro con dos campos. (tipo puntero)
***p es un registro con dos campos (tipo registro)**
(*p).num es una variable simple (tipo entero)
p->num es una variable simple (tipo entero)
&x es la direccion de una variable x, siendo x, por ejemplo int x;
Si deseamos que una variable tipo puntero no apunte a nada, asignamos la palabra reservada NULL (p=NULL)

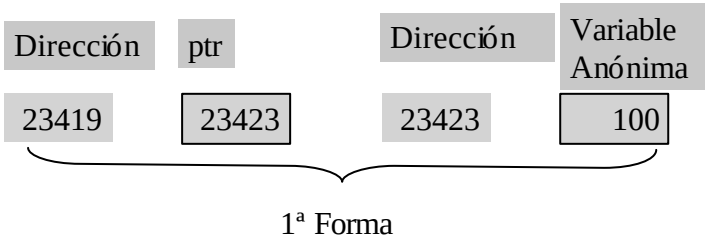
Variables Anónimas

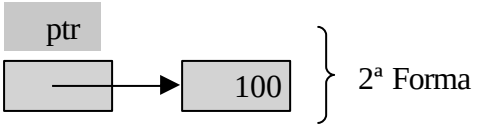
Variables `int *ptr;`

*ptr=100;



Representación Gráfica de las Variables Puntero

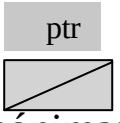




Punteros y Creación de Variables Dinámicas

Inicialización de Punteros:

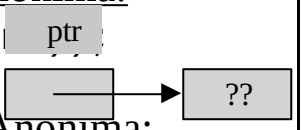
```
Int *ptr;
ptr =  NULL
```



Creación de una Variable Anónima:

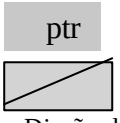
```
ptr=(int *)malloc(sizeof(i
```

(Si no ex existiera memoria libre, devuelve un puntero NULL)



Destrucción de una Variable Anónima:

```
free(ptr);
```



Punteros y Creación de Variables Dinámicas

```
typedef tiporegistro *tipopuntero;
tipopuntero p;
if ((p=(tiporegistro*)malloc(sizeof(tiporegistro))==NULL)
{printf("No hay memoria");
  exit(1);
}
```


Operaciones con Punteros

Dereferenciación: `ptr->PartReal`

Comparación: `ptr1 == ptr2`

Asignación: `ptr1 = ptr2`

Ejemplo:

```
typedef struct { float partereal,partecompleja;}
partescomplejas;
typedef partescomplejas *complejo;
```

```
complejo punt1,punt2;
punt1->partereal=5.0;
punt1->partcompleja=1.2:
punt2=punt1;
```

Operaciones sobre Listas Enlazadas

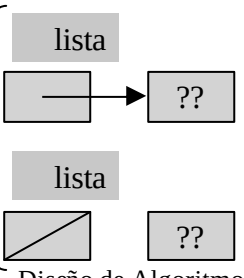
INICIALIZACION

Correcto

```
lista = NULL
```

Erróneo

```
lista=malloc(…….)
 lista =NULL
```



Declaración de listas en C

```
#include <stdlib.h>
typedef struct nodo
{ int dato;
  struct nodo *sig;
} tlista;

tlista *primero;
```

Visualizar una Lista

```
visualizaLista(Tlista *lista)
  {
       Tlista *recorrer;

       recorrer = lista
       WHILE recorrer != NULL
       {   Escribir(recorrer->dato);
           recorrer = recorrer->sig
       }
   }
```


Operaciones básicas sobre listas enlazadas.

Suponiendo:

typedef struct nodo { int dato;
struct nodo *sig; } tiponodo;

typedef tiponodo *tipolista;

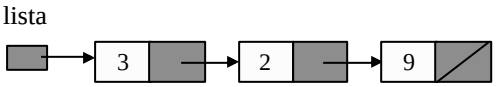
tipolista lista; /*cabeza de la lista*/

tipolista nodo; /*nuevo nodo a insertar*/

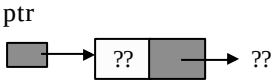
tipolista ptr; /*puntero auxiliar*/

Insertar un Nodo al Principio

Suponiendo:

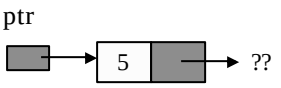



```
1.- ptr=malloc(sizeof(tiponodo));
```

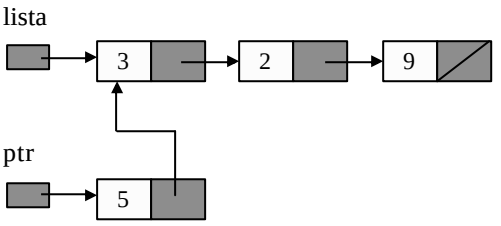


Insertar un Nodo al Principio

```
2.- ptr->dato= 5
```

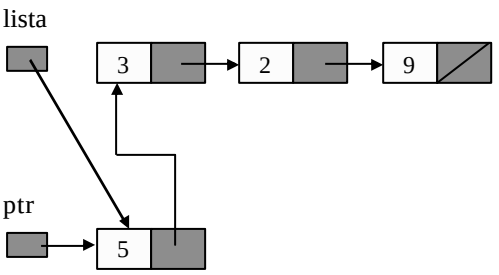



```
3.- ptr->sig= lista
```



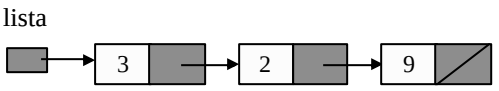
Insertar un Nodo al Principio

```
4.- lista= ptr
```

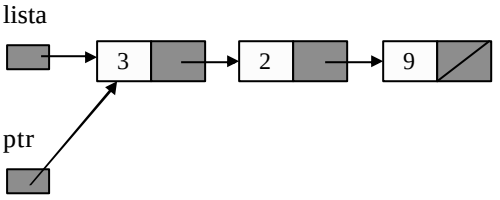


Eliminar el Primer Nodo

Suponiendo:

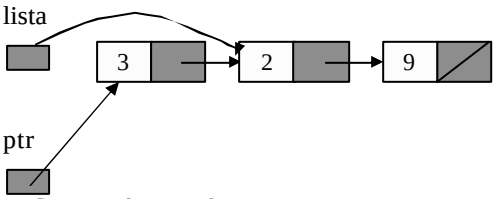



```
1.- ptr = lista
```

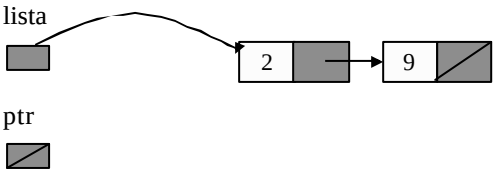


Eliminar el Primer Nodo

```
2.- lista = lista->sig;
```



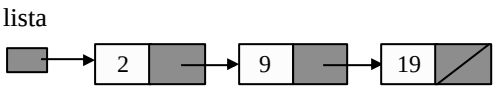

```
3.- free(ptr);
```



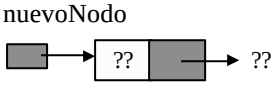
Insertar un Nodo en una Lista Enlazada Ordenada

Partimos de la lista:

Queremos insertar el número: 15

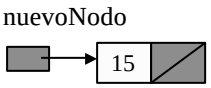



```
1.-
  nuevonodo=malloc(sizeof(tiponodo))
```



Insertar un Nodo en una Lista Enlazada Ordenada

```
2.- nuevoNodo->dato = 15
    nuevoNodo->sig  = NULL
```




```
3.- Algoritmo que inserta el nodo
    en la posición correcta.
```

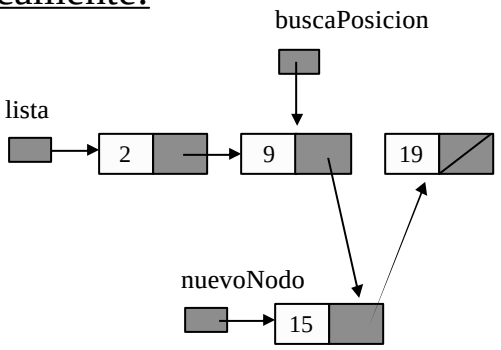

Insertar un Nodo en una Lista Enlazada Ordenada

- Si la lista no está vacía utilizamos unm bucle similar al siguiente:

```
while  ((ptr→sig!=NULL) &&
(nuevonodo→dato > (ptr→sig)→dato)))
 ptr=ptr→sig;
Nodo→sig=ptr→sig;
Ptr→sig=nodo;
```

Insertar un Nodo en una Lista Enlazada Ordenada

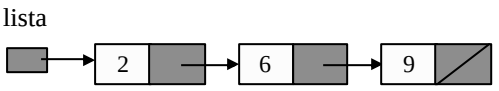
Gráficamente:



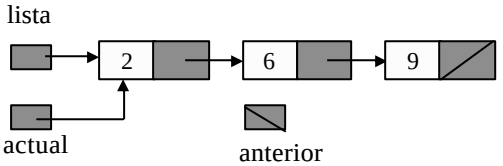
Eliminar un Nodo de una Lista Enlazada

Suponiendo:

Queremos borrar el número: 6
Supondremos que está en la lista.



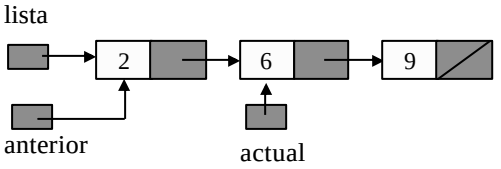

```
1.- actual   = lista
    anterior = NULL
```



Eliminar un Nodo de una Lista Enlazada

2.- Búsqueda del nodo a borrar.

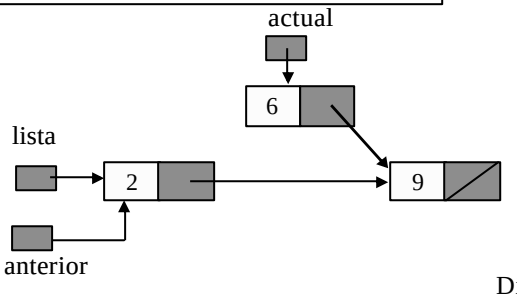
```
while (actual->dato != dato)
{    anterior = actual;
    actual= actual->sig;
}
```



Eliminar un Nodo de una Lista Enlazada

3.- Actualizar los punteros

```
if (anterior ==NULL)
   lista = lista->sig
else
   anterior->sig = actual->sig
```



Eliminar un Nodo de una Lista Enlazada

```
4.- free(actual);
```