Estructuras de Datos Dinámicas

Contenido del Tema

TEMA 2

2.1. Introducción a las estructuras de datos dinámicas.

2.2. Tipo Puntero

2.3. Aplicación: Listas enlazadas.

Introducción a las Estructuras de Datos Dinámicas

Problema

⇩

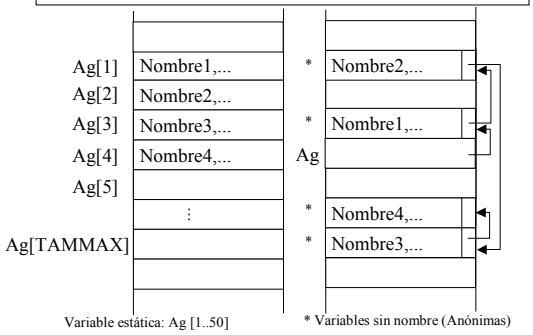
¿Qué sucede si a priori no conocemos la cantidad de espacio de almacenamiento que vamos a precisar?

Solución ⇔ Hacer una previsión??

Ejemplo:

```
Tipos
 TipoPersona TipoPoblacion[1..TAMMAX]
Variables
 TipoPoblacion poblacion
```


Necesidad de Memoria Dinámicas



Variable estática: Ag [1..50]

* Variables sin nombre (Anónimas)

Definición de Puntero

¿Que Es un Puntero?

Variable Estática

Un puntero es una variable que almacena una dirección de memoria

Declaración de Punteros

```
TIPOS
  Tipo *TPtrATipo
VARIABLES
  TPtrATipo ptr
```

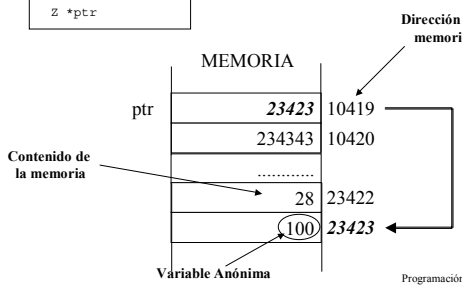
1ª FORMA

```
VARIABLES
  Tipo *ptr
```

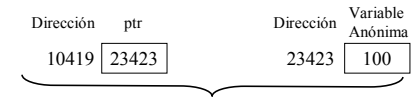
2ª FORMA

Variables Anónimas

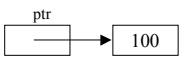
```
VARIABLES
  Z *ptr
```



Representación Gráfica de las Variables Puntero



1ª Forma (Como aparece en memoria)

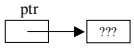
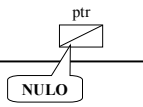


2ª Forma (Representación Gráfica)

Operaciones con Punteros

Operación	Código	Significado
Desreferenciación	`*ptr`	Permite acceder a la variable anónima asociada al puntero *ptr*.
Comparación	`ptr1==ptr2`	Esta comparación será verdadera cuando *ptr1* y *ptr2* apunten a la misma dirección de memoria
Asignación	`ptr1=ptr2` `ptr=NULO`	*ptr1* pasa a apuntar a la misma variable anónima que *ptr2*. NIL indica que no apunta a ninguna dir. de memoria

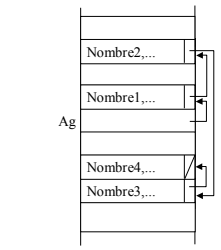
Punteros y Creación de Variables Dinámicas

Operación	Código	Representación Gráfica
Creación de una Variable Anónima	`Asigna (ptr)`	
Destrucción de una Variable Anónima	`Libera (ptr)`	

Operaciones con Punteros

```
TIPOS
  Z* TPtrAZ
VARIABLES
  TPtrAZ ptr1, ptr2, ptr3
INICIO
  Asigna(ptr1)
  Asigna(ptr2)
  ptr3=ptr1
  *ptr1=3
  *ptr2=3
  Libera(ptr3)
  ptr1=NULO
```

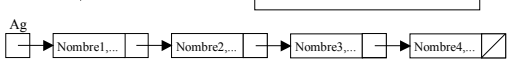

Introducción a las Listas Enlazadas




```
TIPOS
  TNodo *TPuntANodo
REGISTRO  TNodo
     TPersona info
     TPuntANodo sig
FIN
VARIABLES
   TPuntANodo Ag
```

Para acceder a los componentes de un registro desde una variable puntero se utiliza '→'

ag→info ■ (*ag).info



Operaciones de Listas Enlazadas

- Insertar un nodo al principio
- Eliminar el primer nodo
- Insertar un nodo en una lista enlazada ordenada
- Eliminar un nodo en una lista enlazada
- Eliminar todos los nodos de luna lista enlazada
- Visualizar una lista

Insertar un Nodo al Principio

```
ALGORITMO InsPrim(E/S TLista l, E TDato d)
VARIABLES
  TPtrANodo ptr
INICIO
  Asigna (ptr)
  ptr->datdo= 5
  ptr->sig= lista
  lista= ptr
FIN
```


Eliminar el Primer Nodo

```
 ALGORIMO ElimPrim (E/S Tlista l)
  VARIABLE
    TPtrANodo ptr
  INICIO
       ptr = lista
       lista = lista->sig
       Libera(ptr)
  FIN
```


Insertar un Nodo en una Lista Enlazada Ordenada

```
ALGORITMO InsOrd(E/S Tlista l, E Tdato d)
VARIABLES
   TPuntANodo ant, ptr, nuevonodo
INICIO
   Inicializar(nuevoNodo, d)
   BuscarPosicion(lista, ant, ptr, d)
   Enlazar(lista, ant, ptr, nuevo)
FIN
```


Insertar un Nodo en una Lista Enlazada Ordenada

```
ALGORITMO Inicializar(S TPuntANodo nodo, E TDato d)
INICIO

   Asignar(nodo)

   nodo->dato = d

FIN
```


Insertar un Nodo en una Lista Enlazada Ordenada

```
ALGORITMO BuscarPosición(E TLista l, E/S TPtrANodo ret,
                         E/S TPtrANodo av, Tdato d)
INICIO
   ret = NIL
   av = l
   MIENTRAS (av != NULO) Y (av->dato < d) HACER
        ret = av
        av = av->sig
   FIN MIENTRAS
FIN
```


Insertar un Nodo en una Lista Enlazada Ordenada

```
ALGORITMO Enlazar(E/S Tlista l, TPtrANodo ant,
                  TPtrANodo av, TPtrANodo nuevo)
INICIO
   SI ant == NULO ENTONCES     // VA EN PRIMERA POSICIÓN
        nuevo->sig = l
        l =nuevo
   SI NO                       // NO VA EN PRIMERA POSICIÓN
        nuevo->sig = av
        ant->sig = nuevo
   FINSI
FIN
```

Eliminar un Nodo en una Lista Enlazada

```
ALGORITMO ElimNodo(E/S Tlista l, TDato d)
VARIABLES
     TPtrANodo ant, ptr
INICIO
     BuscarPosición(l, ant, ptr, d)
     Suprimir(l, ant, ptr)
FIN
```


Eliminar un Nodo en una Lista Enlazada

```
ALGORITMO Suprimir(E/S Tlista l, TPtrANodo atr, TPtrANodo av)
INICIO
     SI av != NULO ENTONCES  // HEMOS ENCONTRADO EL ELEMENTO
            SI ant == NULO ENTONCES // ES EL PRIMER ELEMENTO
                   l = l->sig
            SI NO
                   ant->sig = ptr->sig
            FINSI
            Libera (ptr)
     FINSI
FIN
```


Eliminar Todos los Nodos de una Lista Enlazada

```
ALGORITMO BorrarLista(ES TLista lista)
INICIO
  MIENTRAS NO Vacía(lista) HACER
        ElimPrim(lista)
  FINMIENTRAS
FIN
```


Visualizar una Lista

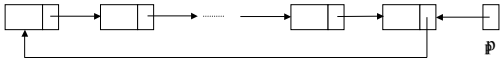
```
ALGORITMO visualizaLista(E TLista lista)
VARIABLES
   TPtrANodo recorrer
INICIO
   recorrer = lista
   MIENTRAS recorrer != NULO HACER // FIN DE LISTA
        Escribir(recorrer->dato);
        recorrer = recorrer->sig
   FINMIENTRAS
FIN
```


Listas enlazadas circulares

- El campo de enlace del último nodo apunta al primer nodo de la lista, en lugar de tener el valor **NIL**
- No existe ni primer ni último nodo. Tenemos un anillo de elementos enlazados unos con otros

Listas enlazadas circulares

- Es conveniente, aunque no necesario, tener un enlace (puntero o índice) al último nodo lógico de la lista. Así podemos acceder facilmente a ambos extremos de la misma
- Una lista circular vacía vendrá representada por un valor **NIL** o **Nulo** en p

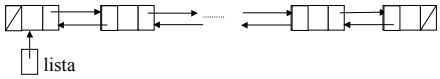


Listas enlazadas circulares

```
ALGORITMO Imprimir(E TLista l)
VARIABLES
    TipoPuntero ptr
INICIO
   ptr = l
   SI (ptr != NULO) ENTONCES
        REPETIR
                Escribir(ptr->elemento)
                ptr = ptr->sig
        HASTA QUE ptr == l
   FINSI
FIN
```

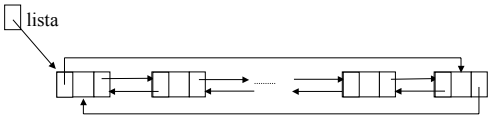

Listas doblemente enlazadas

- Es una lista enlazada en la que cada nodo tiene al menos tres campos:
  - Elemento. El dato de la lista.
  - Enlace al nodo anterior.
  - Enlace al nodo siguiente.
- Los algoritmos para las operaciones sobre listas doblemente enlazadas son normalmente más complicados.
- Pueden ser recorridas fácilmente en ambos sentidos.



Listas doblemente enlazadas

- Una lista doblemente enlazada puede modificarse para obtener una estructura circular de la misma



BIBLIOGRAFIA

- Dale, N. y Weems, C. ***Pascal***. Ed. McGraw Hill, 2ª edición, 1989
- Helman, P., Veroff, R. y Carrano, F. ***Intermediate Problem Solving and Data Structures. Walls & Mirrors***. Ed. The Benjamin/Cummings Publishing, 2ª edición, 1991.
- Joyanes Aguilar, L. ***Fundamentos de Programación. Algoritmos y Estructuras de Datos***. Ed. McGraw Hill, 2ª edición, 1996.
- Tanenbaum, A.M., Augenstein, M.J. ***Data Structures Using Pascal***. Ed. Prentice Hall, 2ª edición, 1986.