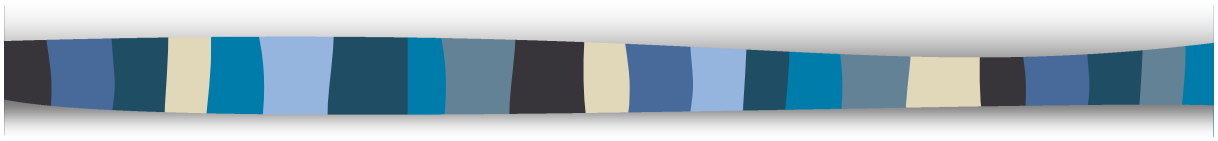
Estructuras de Datos



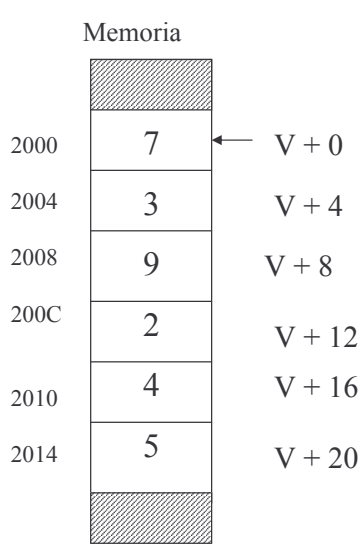
Material Elaborado por el Profesor Ricardo González
A partir de Materiales de las Profesoras
Angela Di Serio
Mariela Curiel

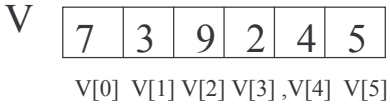
Estructuras de datos

Los programadores de lenguaje ensamblador piensan en término de datos simples: caracteres, enteros, números en punto flotante. Sin embargo, mientras más complejo es el problema que deseamos resolver, las estructuras de datos dejan de ser simples y se hace necesario emplear estructuras más elaboradas como los arreglos, Strings, Clases, listas, pilas y árboles. Cada una de estas abstracciones pueden construirse a partir de elementos más simples como lo son: direcciones de memoria, caracteres, enteros y números en punto flotante

Arreglos de una dimensión (vectores)

Los arreglos son colecciones ordenadas de datos que poseen el mismo tipo y formato de datos en sus elementos constitutivos.

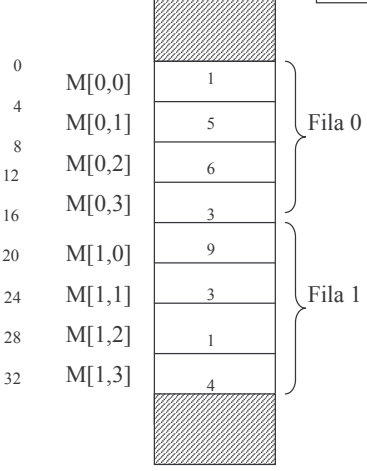




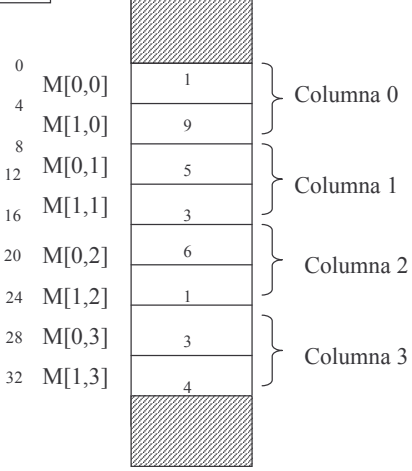

```
move   $t0, $0
lw     $t3, V ($t0)
addi   $t0, $t0, 4
lw     $t4, V ($t0)
```

Arreglos de dos dimensiones (matrices)

	0	1	2	3
0	1	5	6	3
1	9	3	1	4

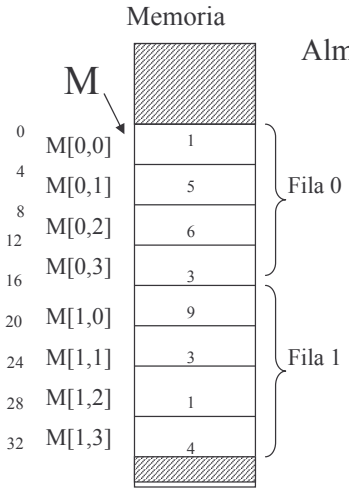


Almacenamiento por filas



Almacenamiento por columnas

Arreglos de dos dimensiones (matrices)



Almacenamiento por filas

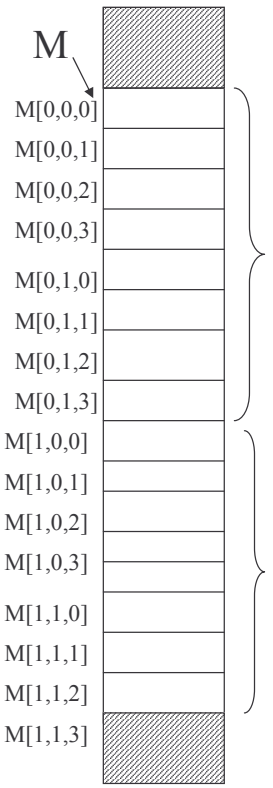
	0	1	2	3
0	1	5	6	3
1	9	3	1	4

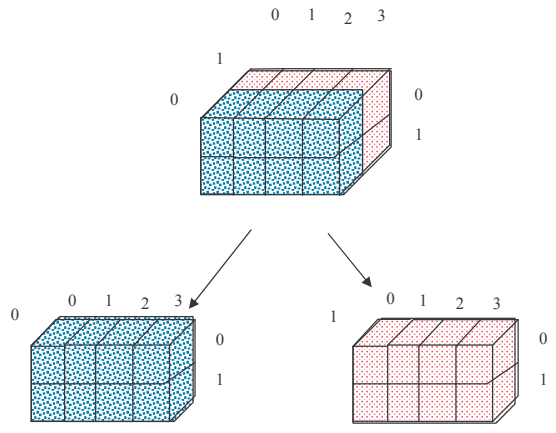
Para calcular el desplazamiento de un elemento con respecto al origen de la matriz, se emplea la siguiente fórmula

Dir efectiva = Dirección de M +
(fila i * número de elementos en una fila * tamaño de un elemento) +
(columna j * tamaño de un elemento)

Die efectiva M[1,2] = Direccion de M + (1 * 4 * 4) + (2 * 4)
= Direccion de M + 24

Matrices multidimensionales





Registros

Item	Numero	Precio unitario	subTotal
Clavos	5	200	1000
Martillo	1	25000	25000
Alicate	2	14000	28000

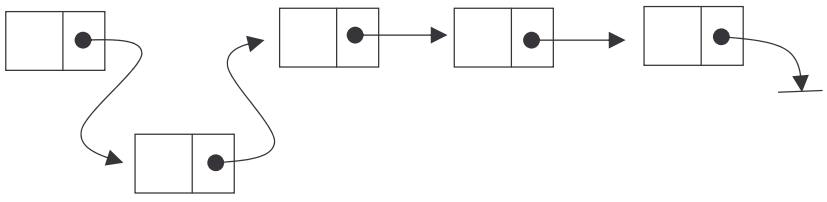
16	4	4	4 bytes
Clavos	5	200	1000
Martillo	1	25000	25000
Alicate	2	14000	28000

Si quiero saber cuantos alicates se vendieron

16 + 4 + 4	24	0
16 + 4 + 4	24	24
16	16	48
		64

Estructuras de datos Dinámicas

Son estructuras que pueden variar en cuanto a la cantidad de sus componentes o a los tipos de datos que las componen.



Objetos

Para los propósitos de esta clase consideraremos que un objeto es una abstracción en la que se especifica una estructura de datos y las operaciones que permiten operar con dicha estructura de datos.

La creacción de la estructura en sí, se realiza con la llamada al constructor de la clase (new), en este momento es que se reserva espacio de memoria para almacenar los datos que contiene el objeto.

La referencia a los objetos se realiza al seguir apuntadores a las direcciones de memorias asignadas a estas estructuras.

Ejemplo 1 de un objeto

Clase Point

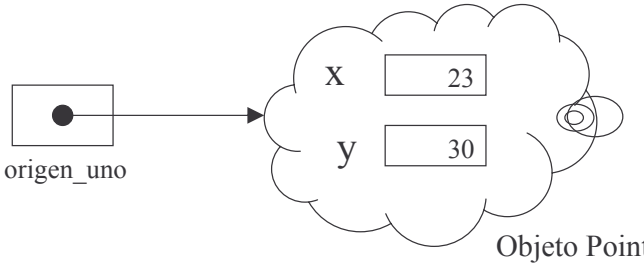
```
Public clas Point {
   public int x=0;
   public int y=0;
   public Point(int x, int y) {
      this.x = x;
      this.y = y;
   }
}
```

Point origen_uno =>

origen_uno

Cuando hacemos un new lo que ocurre es:

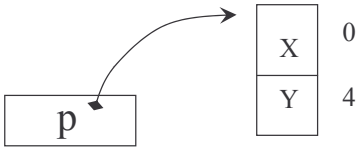
Point origen_uno = new Point(23, 30);



Ejemplo 1 de un objeto

Clase Point

Objeto p



p = new Point(a, b)

```
lw      $t3, a
lw      $t4, b
li      $a0, 8
li      $v0, 9
syscall
move    $t5, $v0
sw      $t3, 0 ($t5)
sw      $t4, 4 ($t5)
sw      $t5, p
```

p.getY()

```
lw      $t2, p
lw      $t3, 4 ($t2)
move    $v0, $t3
jr      $ra
```

sbrk	9	$a0 = amount	address (in $v0)

sbrk returns a pointer to a block of memory containing *n* additional bytes.

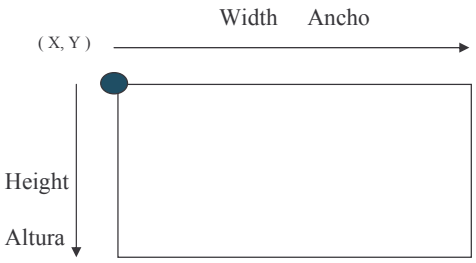
Syscall

sbrk returns a pointer to a block of memory containing *n* additional bytes.

Service	System call code	Arguments	Result
print_int	1	$a0 = integer	
print_float	2	$f12 = float	
print_double	3	$f12 = double	
print_string	4	$a0 = string	
read_int	5		integer (in $v0)
read_float	6		float (in $f0)
read_double	7		double (in $f0)
read_string	8	$a0 = buffer, $a1 = length	
sbrk	9	$a0 = amount	address (in $v0)
exit	10		
print_char	11	$a0 = char	
read_char	12		char (in $a0)
open	13	$a0 = filename (string), $a1 = flags, $a2 = mode	file descriptor (in $a0)
read	14	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars read (in $a0)
write	15	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars written (in $a0)
close	16	$a0 = file descriptor	
exit2	17	$a0 = result	

Ejemplo 2 de un objeto

Rectángulo

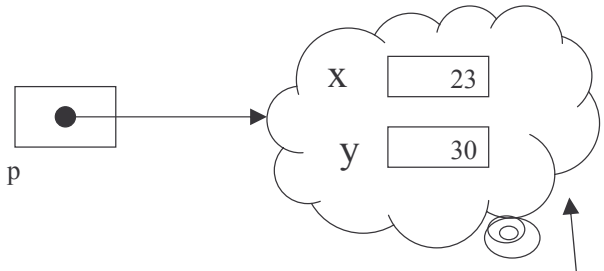


Ejemplo 2 de un objeto

Clase Rectangle

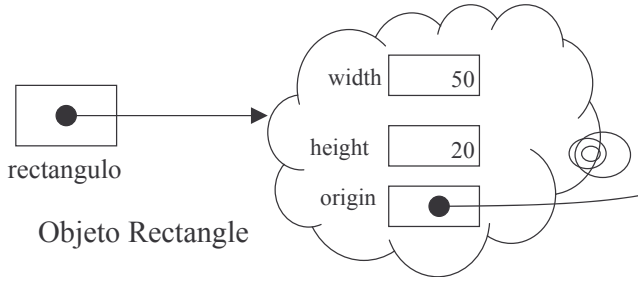
```
Public class Rectangle{
   public int width;
   public int height;
   public Point origin;

   public Rectangle(Point
     p,int x, int y) {
      origin = p;
      width = x;
      height = y;
   }
}
```



Cuando hacemos un new lo que ocurre es:

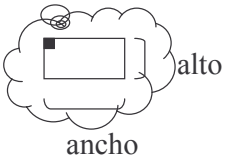
Rectangle rectangulo = new Rectangle(p, 50, 20);



Objeto Rectangle

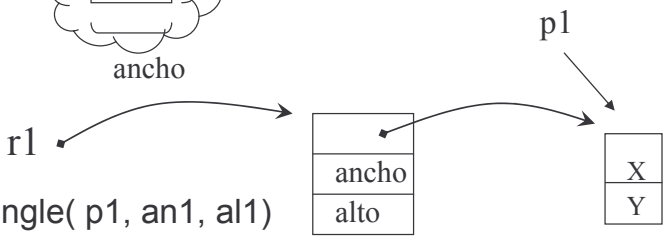
Ejemplo 2 de un objeto

Clase Rectangle



Rectangle(Point p, int n, int l)

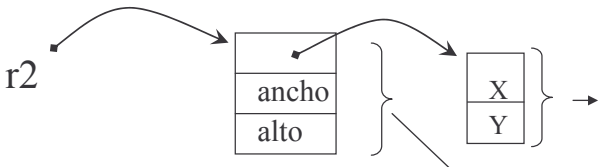
Objeto r1



r1 = new Rectangle(p1, an1, al1)

```
lw      $t2, p1
lw      $t3, an1
lw      $t4, al1
li      $a0, 12
li      $v0, 9
syscall
move    $t5, $v0
sw      $t2, 0 ($t5)
sw      $t3, 4 ($t5)
sw      $t4, 8 ($t5)
sw      $t5, r1
```

Ejemplo 2 de un objeto

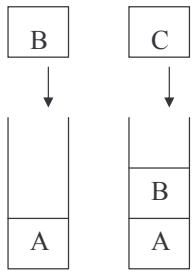


r2 = new Rectangle (int x, int y, int al, int an)

```
lw      $t0, x
lw      $t1, y
lw      $t2, an
lw      $t3, al
li      $a0, 8
li      $v0, 9
syscall
move    $t5, $v0
sw      $t0, 0 ($t5)
sw      $t1, 4 ($t5)
li      $a0, 12
li      $v0, 9
syscall
move    $t6, $v0
sw      $t5, 0 ($t6)
sw      $t2, 4 ($t6)
sw      $t3, 8 ($t6)
jr      $ra
```

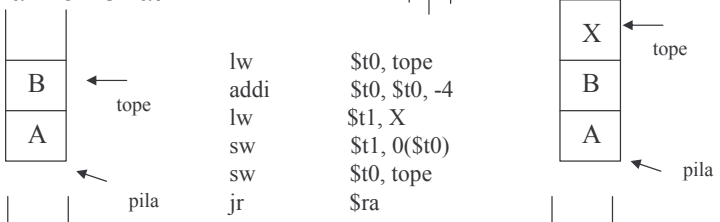
Pilas

Una pila es una estructura de datos que permite implementar la disciplina FILO (First In Last Out) Para utilizar la pila se pueden emplear dos valores diferentes, uno para indicar la direccion o el comienzo de la pila y otro para el tope o último elemento incluido en la pila.

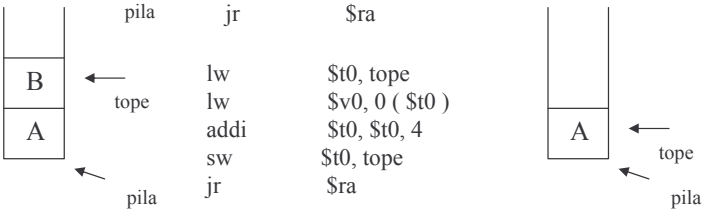


Si la pila crece de las direcciones altas a las direcciones bajas de la memoria.

apilar (X)

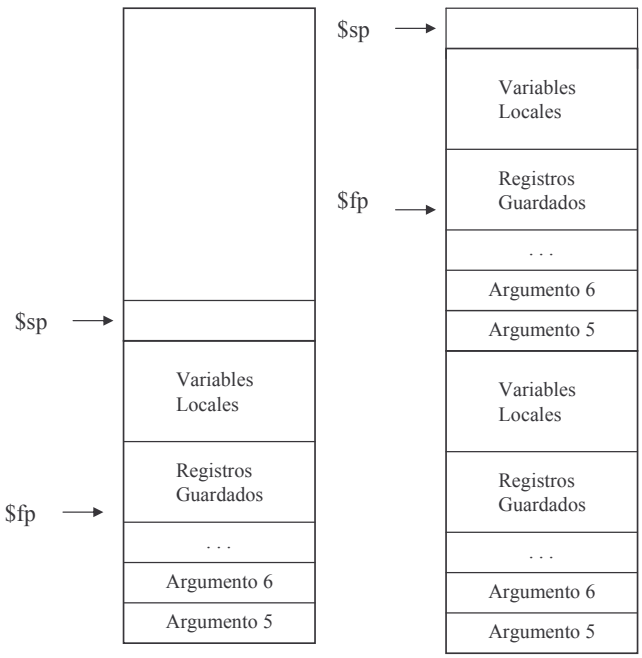


desapilar ( )



Pilas

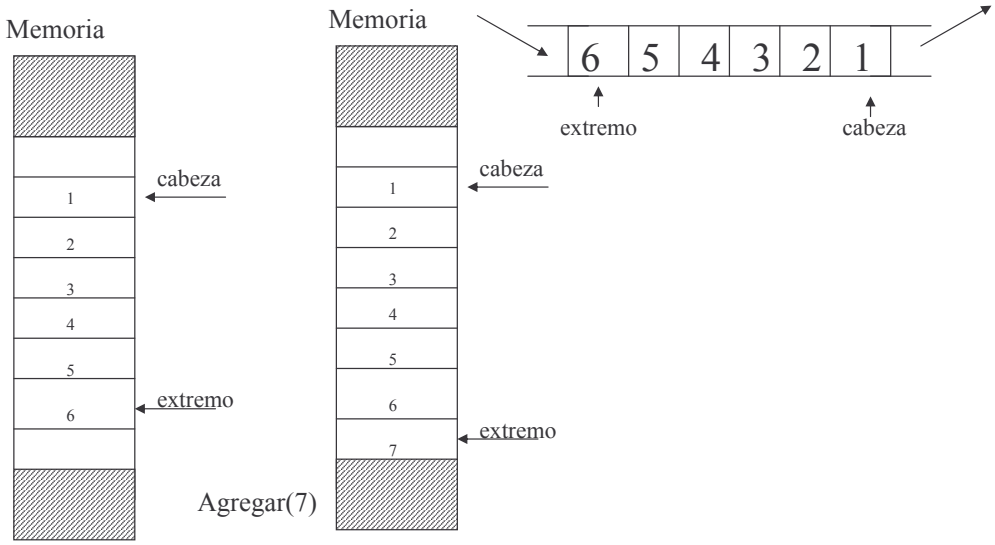
Si los datos a almacenar en la pila son de longitud variable y/o no se sabe a priori el tamaño de los nodos a incluir en la pila, es posible que se requieran otras estructuras para manejar la pila.



Colas

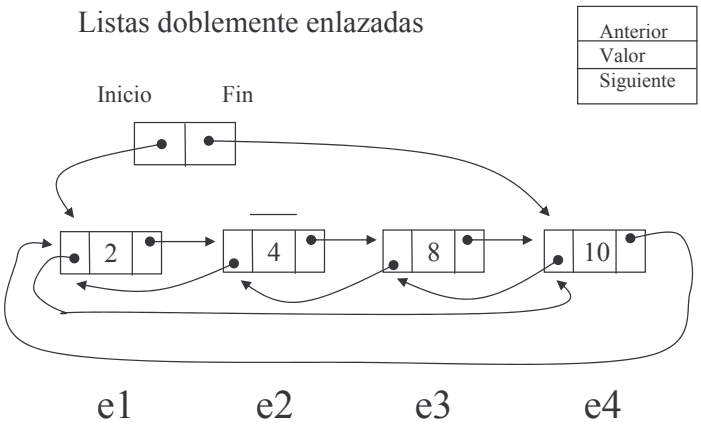
Una cola es una estructura de datos que permite implementar la disciplina FIFO (First In First Out)

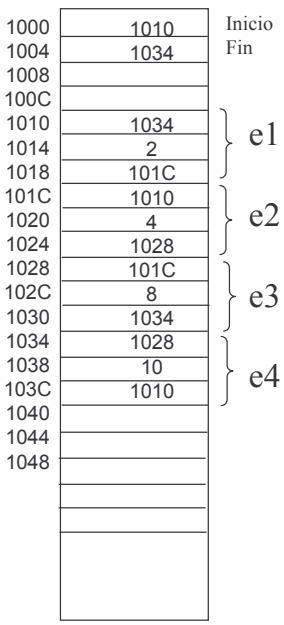
Para utilizar la cola se emplean al menos dos valores diferentes, uno para la cabeza de la cola y otro para el final, cola o extremo de la cola.



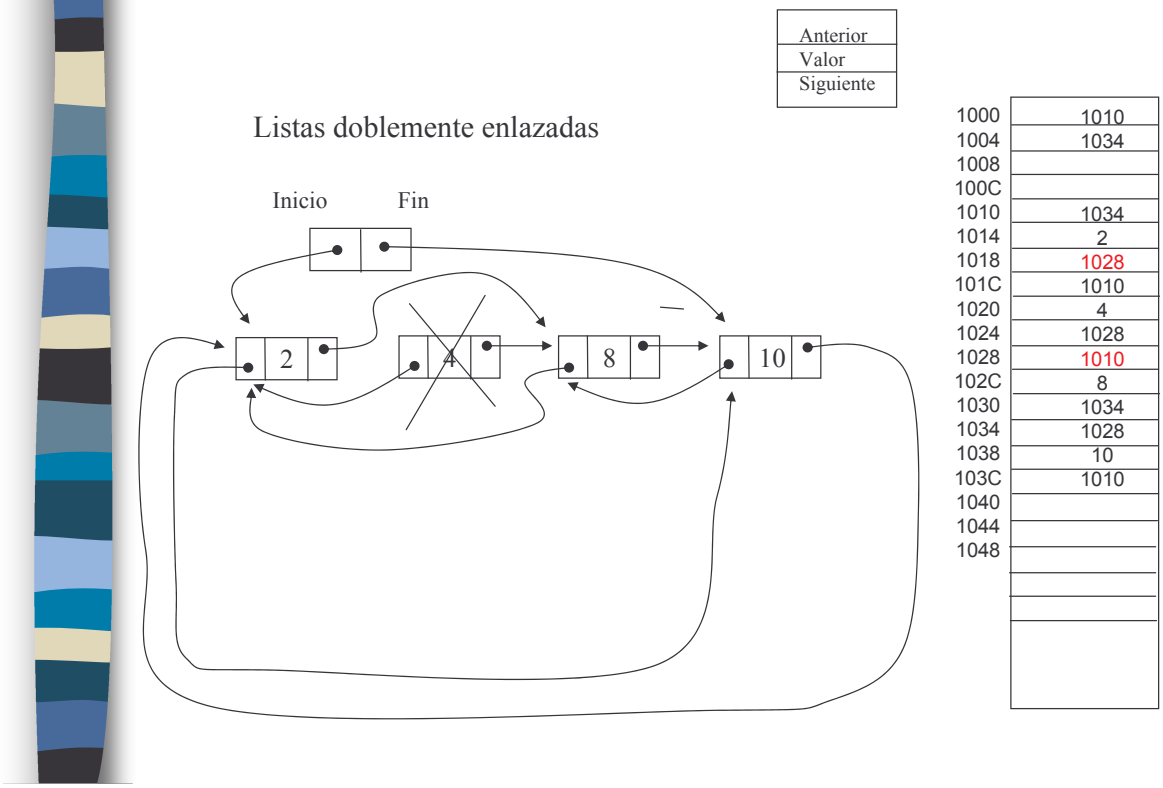
Estructuras de datos Dinámicas

Listas doblemente enlazadas



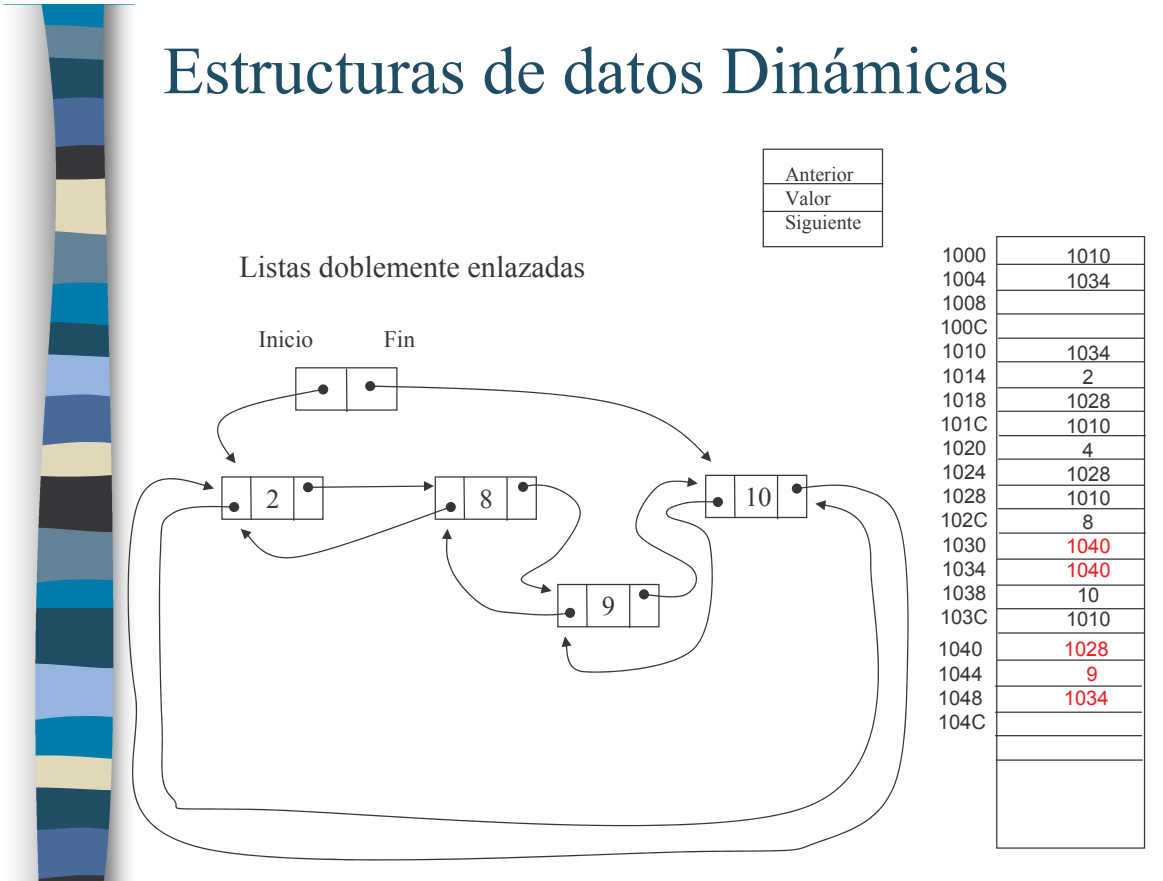


Estructuras de datos Dinámicas



1000	1010
1004	1034
1008	
100C	
1010	1034
1014	2
1018	1028
101C	1010
1020	4
1024	1028
1028	1010
102C	8
1030	1034
1034	1028
1038	10
103C	1010
1040	
1044	
1048	

Estructuras de datos Dinámicas



1000	1010
1004	1034
1008	
100C	
1010	1034
1014	2
1018	1028
101C	1010
1020	4
1024	1028
1028	1010
102C	8
1030	1040
1034	1040
1038	10
103C	1010
1040	1028
1044	9
1048	1034
104C	