

Listas enlazadas en lenguaje C

El Problema

Generar una lista de varios elementos del mismo tipo

```
typedef struct sData {  
    unsigned int r1;  
    unsigned int r2;  
} tData;
```

Solución Típica

Vector de estructuras

```
tData V[500];
```

Pero...

- * Debo conocer la cantidad máxima antes de la compilación
- * La cantidad de elementos de la lista es fija
- * Reserva memoria que puede no usarse nunca
- * Inadecuado para grandes bloques de datos

Solución Típica

Bloque de Memoria

```
int cant = 500;           // Tiempo de ejecución
tData * V;
V = (tData *) malloc(cant * sizeof(tData));
```

Pero...

- ~~* Debo conocer la cantidad máxima antes de la compilación.~~
- ~~* La cantidad de elementos de la lista es fija~~
- * Reservo memoria que puede no usarse nunca
- * Inadecuado para grandes bloques de datos
- * No puedo desconocer la cantidad de datos
- * Cambiar la cantidad implica mover bloques de memoria

Solución Típica

Bloque de Índices

```
int cant = 500;                // Tiempo de ejecución
tData ** V;
V = malloc(cant * sizeof(tData *));
v[0] = (tData *) malloc(sizeof(tData));
v[1] = (tData *) malloc(sizeof(tData)); ...
```

Pero...

- ~~* Debo conocer la cantidad máxima antes de la compilación.~~
- ~~* La cantidad de elementos de la lista es fija~~
- * Reserva memoria que puede no usarse nunca
- ~~* Inadecuado para grandes bloques de datos~~
- * No puedo desconocer la cantidad de datos
- * Cambiar la cantidad implica mover bloques de memoria

Entonces?

Necesitamos una solución que:

- * **Nunca** reserve memoria de sobra
- * Pueda **expandir o reducir dinámicamente** la lista
- * **Libere memoria** cuando ya no la utilice
- * Reserve memoria en **bloques atómicos dispersos**
- * Sea **genérica** en su implementación

Porqué?

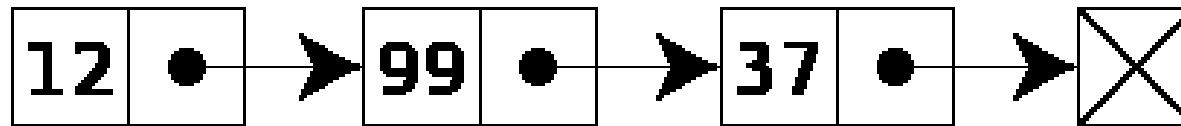
*Un poco de matemática aproximada.
Queremos buffererar 5 segundos de video en HD*

1920px * 1080px	=	2.073.600 px/frame
2.073.600 * 24 bits	=	49.766.400 bits/frame
49.766.400 / 8 bits	≈	6 MB/frame
6MB * 25 frames/seg	≈	150 MB/seg
150MB * 5seg	≈	750 MB/5seg

750MB contiguos en memoria?

Lista Simplemente Enlazada

(o Singly-linked list)



- * Cada bloque es un nodo.
- * Cada nodo contiene un espacio de datos y un vínculo
- * El vínculo (o puntero) apunta al siguiente nodo.
- * El último nodo apunta a NULL como siguiente.

Como implementar una lista simplemente enlazada?

```
struct sData {  
    unsigned int r1;  
    unsigned int r2;  
    struct sData * next;  
};
```

MAL

Mezcla datos con vínculo.

```
struct sNode {  
    tData * data;  
    struct sNode * next;  
};
```

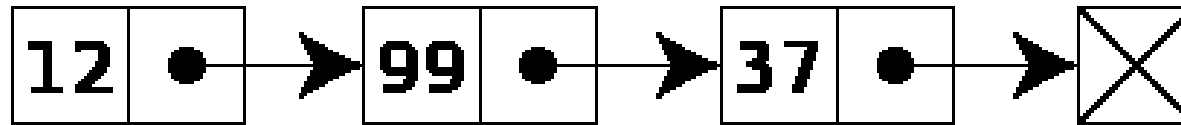
BIEN

Separa datos de vínculo.

```
struct sNode {  
    void * data;  
    struct sNode * next;  
};
```

MEJOR

La misma lista sirve para cualquier tipo de dato!



Pero...

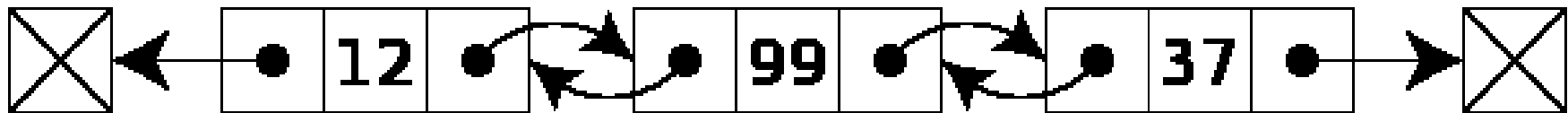
- ~~* Debo conocer la cantidad máxima antes de la compilación.~~
- ~~* La cantidad de elementos de la lista es fija~~
- ~~* Reserva memoria que puede no usarse nunca~~
- ~~* Inadecuado para grandes bloques de datos~~
- ~~* No puedo desconocer la cantidad de datos~~
- ~~* Cambiar la cantidad implica mover bloques de memoria~~



Aunque...

- * Recorrer la lista de atrás para adelante es engorroso.
- * Agregar elementos en el medio de la lista es ineficiente.

Lista Doblemente Enlazada (o Doubly-linked list)



Cada nodo ahora tiene dos vínculos:

- * Un puntero al nodo anterior
- * Un puntero al nodo siguiente
- * Comienzo y final se señalizan con NULL

Como implementar una lista doblemente enlazada?

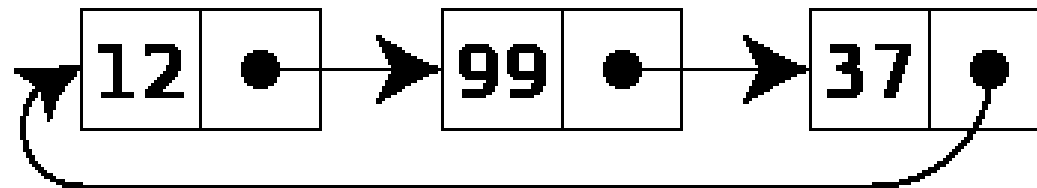
```
struct dNode {  
    void          * data;  
    struct dNode * prev;  
    struct dNode * next;  
};
```



- * La lista se puede recorrer en ambos sentidos
- * Intercalar elementos es más sencillo y eficiente.

Que otros tipos de listas enlazadas existen?

- * Listas circulares



- * Pilas o Stacks (LIFO)

- * Colas (FIFO)

- * Árboles binarios, heaps

- * Etc...

Fuentes

http://en.wikipedia.org/wiki/Linked_list

<http://cslibrary.stanford.edu/103/>

<http://cslibrary.stanford.edu/105/>

[http://es.wikipedia.org/wiki/Lista_\(inform%C3%A1tica\)](http://es.wikipedia.org/wiki/Lista_(inform%C3%A1tica))

The C Programming Language, 2nd Edition

Brian Kernighan, Dennis Ritchie

RTFM (Read the *fantastic* manual)

Material disponible en

<https://github.com/sergiokas/c>

