Punteros, listas ligadas y C

Alonso Carrasco:
cristian.carrasco@uc.cl

¿Que es un puntero?

Un puntero es una variable que guarda la posición de memoria de otra variable.

¿Que es un puntero?

Digamos que X es int

int X = 5;

Luego podemos guardar obtener y guardar su posición de memoria.

la expresión **&X** nos devuelve la posición en memoria de la variable a.

si X está en la posición de memoria 100, entonces &X va a retornar 100.

¿Que es un puntero?

Si queremos guardar la posición en memoria de **X** necesitamos un puntero, llamémosle **p**. como **X** es de tipo **int**, entonces p debe ser de tipo **int*.**

```
int X = 5;
```

```
int* p = &X;
```

Ahora p guarda la posición de memoria de X. Decimos que p apunta a X.

¿Que es un puntero?

Para guardar la posición de memoria de una variable de tipo *type,* debemos usar una variable de tipo *type*.*

***type* X = ???;**

***type** p = &X;**

Decimos que el puntero p apunta a X.

¿Que es un puntero?

un puntero nos permite **obtener el valor** de una variable y también nos permite **modificarla**.

```
int  X = 5;
```

```
int* p = &X;
```

(obtener)

```
int X_copy = *p;
```

(X_copy vale 5 ahora)

(modificar)

```
*p = 10;
```

(X ahora vale 10)

¿Que es un puntero?

En resumen, un puntero nos permite tener acceso y control de una variable.

Eficiencia en memoria. Todos los punteros gastan la misma cantidad de memoria que es de 8 bytes. (en vez de pasar para todos lados una variable completa, podemos solo pasar el puntero a dicha variable y podemos ser más eficientes)

¿Que es un puntero?

EDD entendiendo punteros

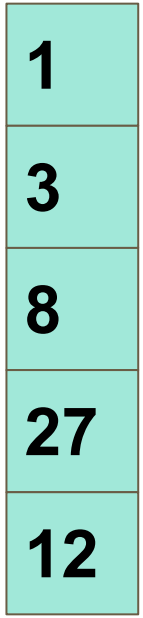
EDD sin entender punteros

2 Tipos de listas (que nos interesan)

- Array
- lista ligada

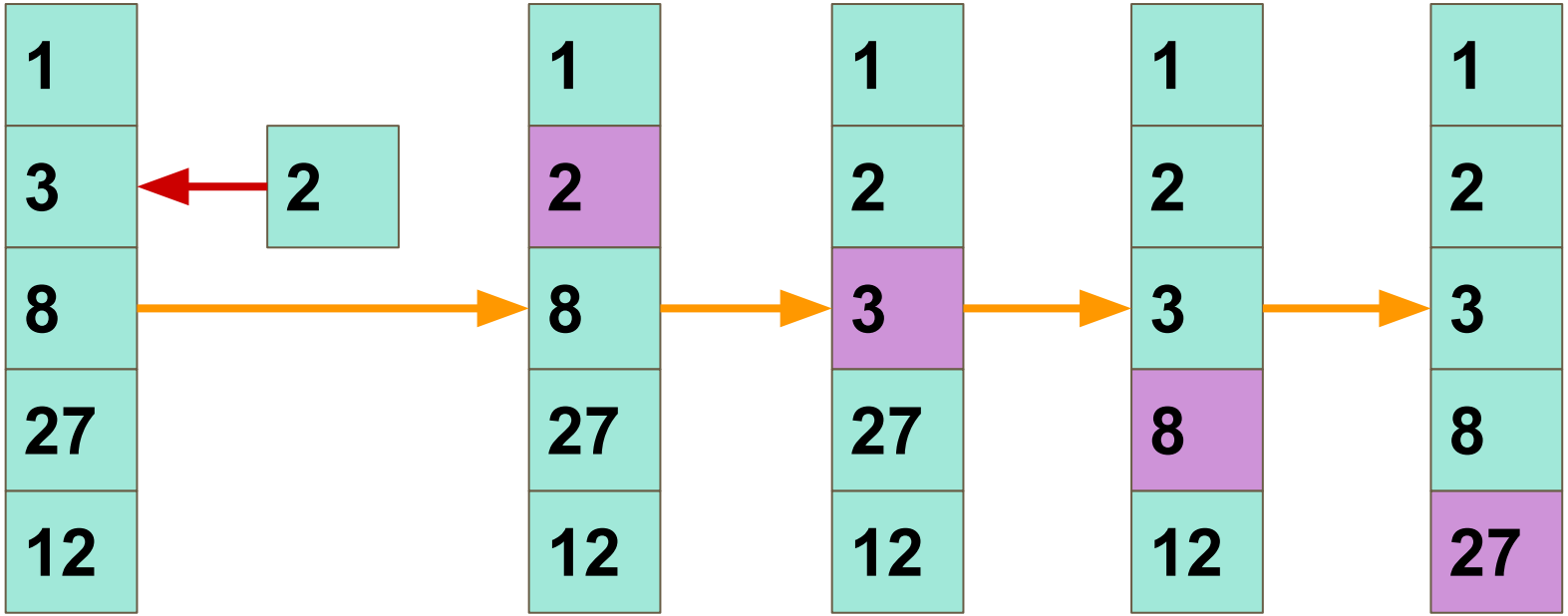
Array

- Un array es una lista donde todos los datos están consecutivos en memoria
- Acceder a un elemento es O(1) según la posición
- **Tienen un largo definido (inmutable)!!!**
- Son difíciles de alterar, hay que mover todos los elementos de la lista para insertar o quitar algo



Array

- insertemos un 2 en la segunda posición

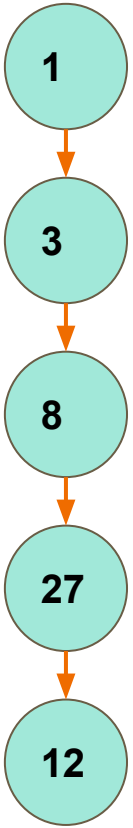


Array

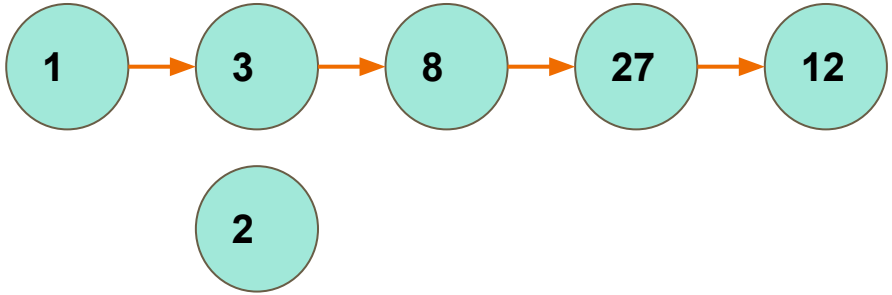
- EL LARGO DE UN ARRAY ES INMUTABLE!!!

Lista ligada

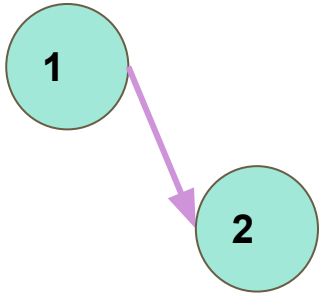
- Estructura de datos organizada en nodos
- cada nodo permite encontrar al nodo siguiente
- cada nodo guarda un valor de la lista
- largo variable
- la inserción es más fácil
- acceder a un elemento específico es costoso

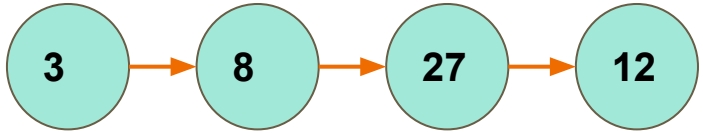


Inserción lista ligada



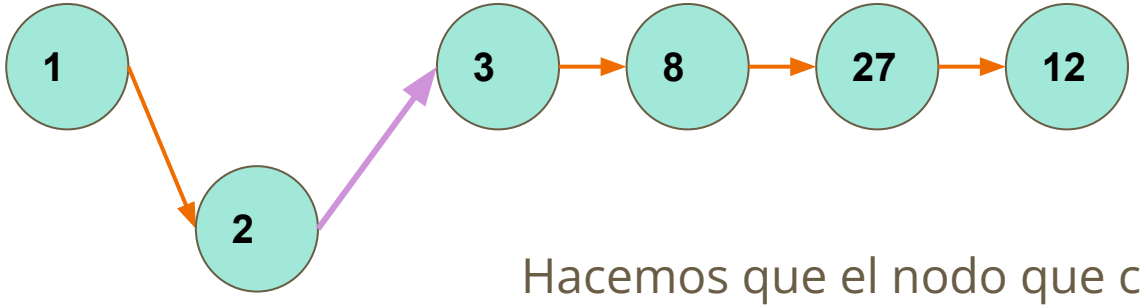
Inserción lista ligada





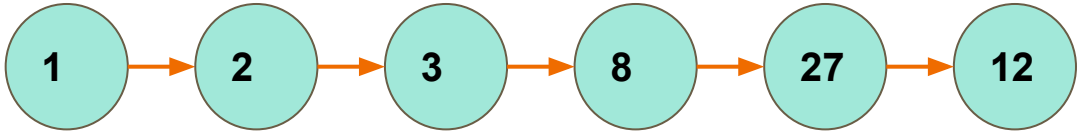
Hacemos que el nodo que contiene al 1 ahora apunte al nodo que contiene al valor 2.

Inserción lista ligada



Hacemos que el nodo que contiene al valor 2 apunte al nodo de valor 3

Inserción lista ligada



- Finalmente insertamos el nodo con valor 2 a la lista ligada

- La lista ligada es mucho más mutable que un array y no tiene largo determinado. Va a depender de la situación que tipo de lista nos conviene usar.

¿Como se modela la lista ligada en C?

Modelando una matriz en C

¿Qué es un matriz?

Una matriz es una lista de listas

vamos a modelar la matriz usando arrays

Recordemos lo que es un array en C

Un array en C es una sección continua de memoria que contiene structs/tipos consecutivos en esta memoria.

Los arrays en C son identificados por un puntero que apunta al primer elemento del array!!!!

Ejemplos

Este es un ejemplo de un array de ints. lo marcado en naranjo es como se vería la memoria del computador. Observe que los ints están consecutivos en memoria

tipo	variable	posición en memoria
int	int_1	101
int	int_2	102
int	lint_3	103
…	…	…
int	int_10	110

Ejemplos

Para identificar este array utilizamos un puntero de forma que éste apunte a la posición en memoria de int_1. se tiene que &int_1 = 101. En código:

int* Array = &int_1;

(& retorna la posición en memoria de una variable)

tipo	variable	posición en memoria
int	int_1	101
int	int_2	102
int	lint_3	103
…	…	…
int	int_10	110

Ejemplos

int lista_int[10];

Esto es un array de 10 ints declarado en el stack. la variable **lista_int** es de tipo **int*. Dicho puntero apunta al primer elemento del array y es lo que identifica al array.**

int* lista_int_2 = malloc(sizeof(int) * 10);

Esto también es un array de ints, pero está declarado en el heap (malloc).

Guardando arrays

(Ejemplo con ints)

En este caso, si nosotros tenemos n Arrays (de cualquier largo), para acceder a cada array necesitamos su puntero que lo identifica.

```
int lista_1[10];
int lista_2[10];
...
int lista_n[10];
```

Guardando arrays

Entonces, para guardar estos n arrays necesitamos guardar los punteros que identifican a cada array. Esto lo podemos hacer con un array guarde a los punteros:

lista_1;
lista_2;
...
lista_n;

Guardando arrays

Las variables:

lista_1;
lista_2;
...
lista_n;

Son de tipo int*. y para guardar un array de estas necesitamos un array de int*.

Guardando arrays

El array se verá de la siguiente forma:

tipo	variable
int*	lista_1
int*	lista_2
int*	lista_3
…	…
int*	lista_n

Guardando arrays

El identificador de este nuevo array es un puntero que apunta al primer elemento del array, que será lista_1. como lista_1 es de tipo int*, entonces este puntero será de tipo int. Se declara así.**

int* Matriz[]

tipo	variable
int*	lista_1
int*	lista_2
int*	lista_3
…	…
int*	lista_n