Estructuras Basicas

¿Cuáles estructuras básicas?

- Arreglos (Arrays)
- Listas Ligadas (Linked Lists)
- Stacks
- Queues

Relevancia

En los lenguajes OOP más comunes, es normal que existan clases/estructuras que permiten almacenar datos para operar sobre ellos de una forma particular.

En Python, existe la clase List que permite obtener elementos por índice, insertar, realizar appends. Todo sin necesidad de saber el tamaño de los datos de antemano

Relevancia

En los lenguajes OOP más comunes, es normal que existan clases/estructuras que permiten almacenar datos para operar sobre ellos de una forma particular.

En Python, existe la clase List que permite obtener elementos por índice, insertar, realizar appends. Todo sin necesidad de saber el tamaño de los datos de antemano

En C **NO existe esto,** sino que todo en C son solo relaciones entre punteros de memoria. Por lo que para utilizar List implicaría implementarlo desde 0 lo que es sumamente ineficiente y e innecesariamente complejo para el contexto de este curso

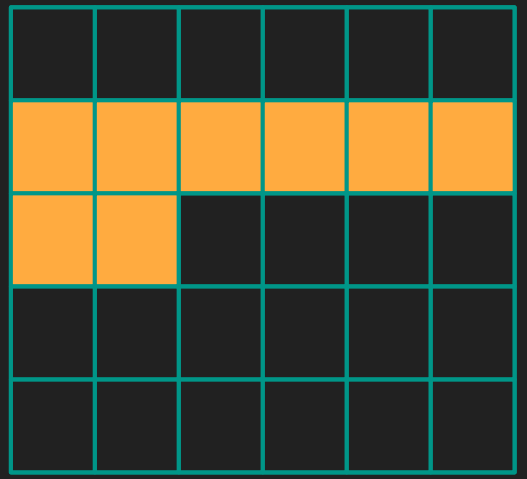
1. Arrays

El Array es la estructura más básica para relacionar datos. Consiste en **N** espacios consecutivos en memoria

int array[7] -> 7 espacios consecutivos en memoria stack (la variable array es un puntero)

MyStruct* structList = calloc(7, sizeof(..)) ->

7 espacios consecutivos en memoria heap



1. ¿Como funciona por detras?

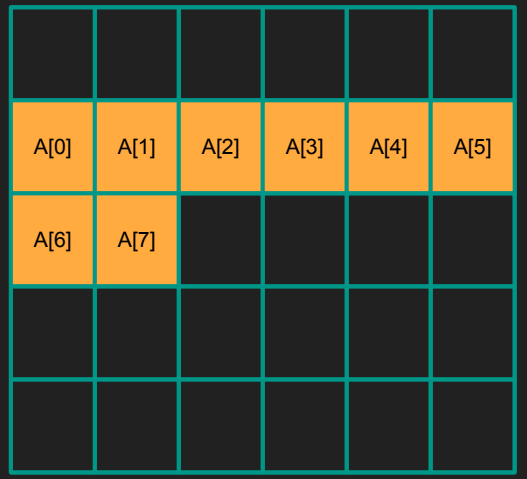
Un array siempre consistirá de un puntero que *apunta* al primer elemento.

int array[7] -> es el puntero a A[0].

Si quiero acceder a A[i], C por detrás realiza la operación: ***array + i * sizeof(int).***

Osea, se mueve i direcciones de memoria.

Ojo: C no sabe el tamaño, por lo que se podria acceder a A[1000], y la instrucción se ejecutara



1. La gran desventaja

Los Arrays tienen un problema!

Al ser celdas de memorias consecutivas. No se puede agrandar directamente (Ya que podrían existir otros datos en la posición N+1).

Para agrandarlo es necesario crear otro array, y copiar los datos hacia dicho array

1. Complejidad de Búsqueda? De Inserción?

- La búsqueda con índice es O(1)! Ya que solo requiere el índice para acceder al dato directamente

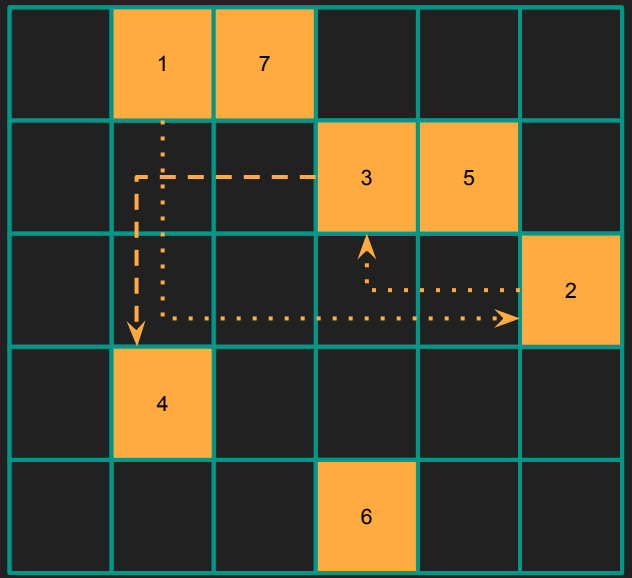
- Si deseo insertar un dato nuevo sobre el tamaño. La inserción es O(n). Porque requiere copiar el array completo y luego agregar el dato.

- Se puede pedir un array sobredimensionado. Pero implicaría que esa memoria esté en uso, aunque realmente no contenga nada.

2. Linked Lists

¿Qué pasa si insertamos muchos datos, pero buscamos poco?

- Las LL son una struct donde cada elemento conoce el elemento anterior o el siguiente. (O ambos)
- Se almacenan en variables solo el primer y último elemento para tener control sobre toda la lista
- Para recorrer es necesario avanzar por cada elemento y continuar al siguiente en la cadena



2. Linked Lists

- El siguiente codigo va creando nuevos elementos y los conecta. Generando una “cadena” de elementos.

```c
typedef struct my_struct_t {
  int value;
  struct my_struct_t* next;
  struct my_struct_t* prev;
} MyStruct;

int main(){
 MyStruct* head;
 MyStruct* tail;
 MyStruct* curr;

 head = calloc(...);
 tail = calloc(...);

 head->next = tail;
 tail->prev = next;

 for(int i=0; i<10; i++) {
   curr = calloc(...);
   tail->next = curr;
   curr->prev = tail;
   tail = curr;
 }
  return 0;
}
```

2. La gran desventaja

Las linked lists tienen una desventaja clave!

Para buscar un elemento, requiere iterar por la lista. Ya que solo se tiene acceso al primer y al último elemento.

1. Complejidad de Búsqueda? De Inserción?

- La búsqueda con índice es O(n)

- La complejidad de inserción es O(1)! ya que solo implica agregar el elemnto al final de la lista

3. Variaciones de Listas ligadas

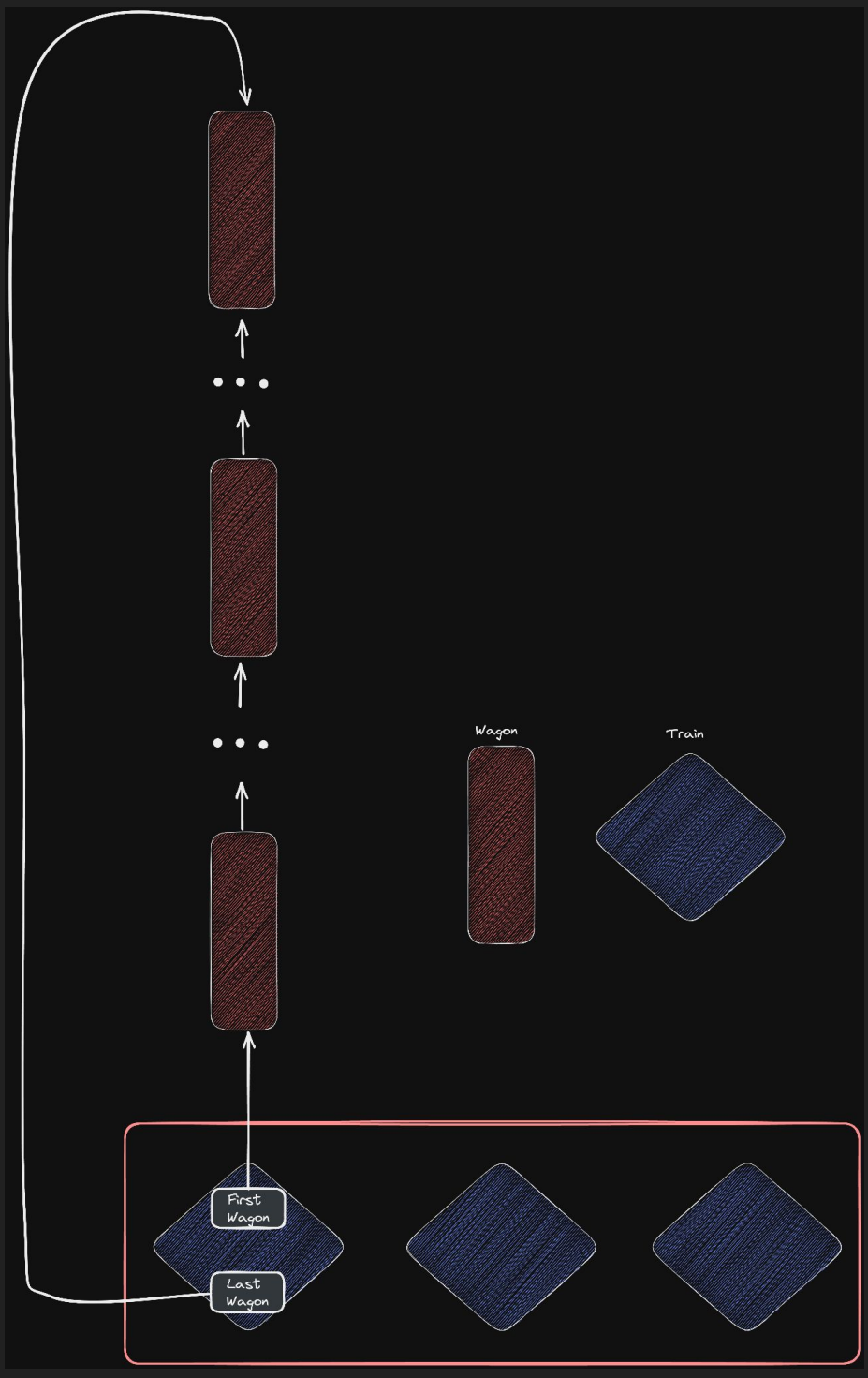
- Una LL donde solo opera sobre el primer elemento (Se inserta en el Head ó se extrae el Head). Se llama Stack

- Una LL donde se extrae el primer elemento y se inserta solo al final. Se llama Queue

Ejemplo!

Vamos a modelar una estacion de trenes.

Tenemos N trenes. y Cada tren posee una cantidad variable de vagones que pueden ser conectados, desconectados o transferidos entre trenes.


Wagon
Train
First Wagon
Last Wagon

Demo!