UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA

# Welcome to
# Algorithms and Data Structures!
# CS2100

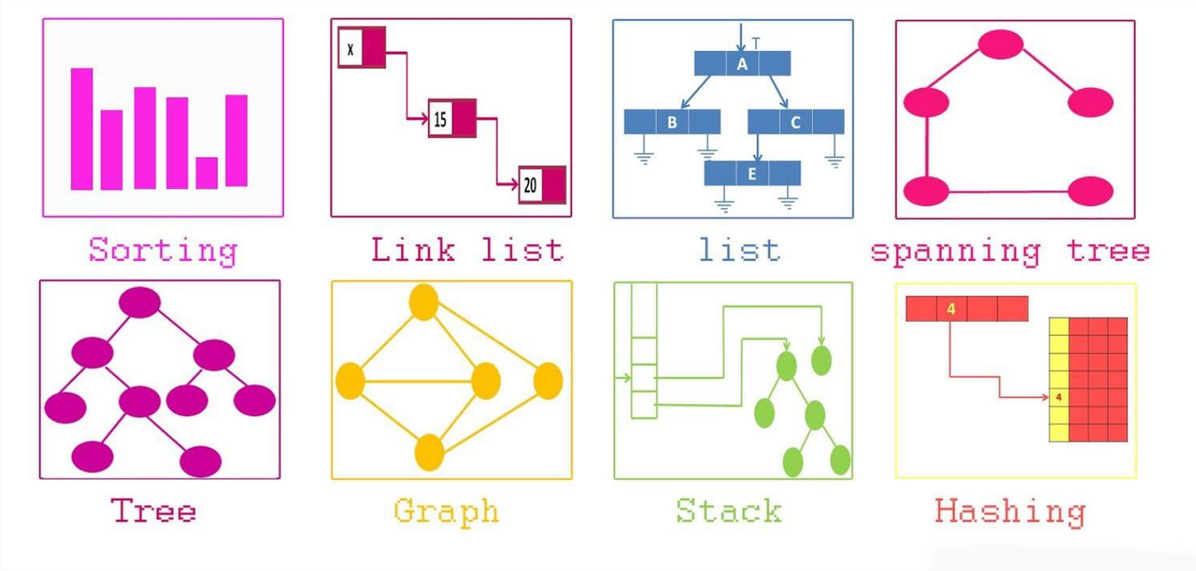
# Arreglos

# ¿Qué es una estructura de datos?

Son colecciones que mantienen diferentes relaciones entre los datos que almacenan. Dichas estructuras permiten un eficiente acceso y modificación de dichos datos

Cada estructura de datos tiene un uso para diferentes problemas. Por ejemplo, una estructura de datos puede utilizarse para guardar información de personas en base a sus nombres

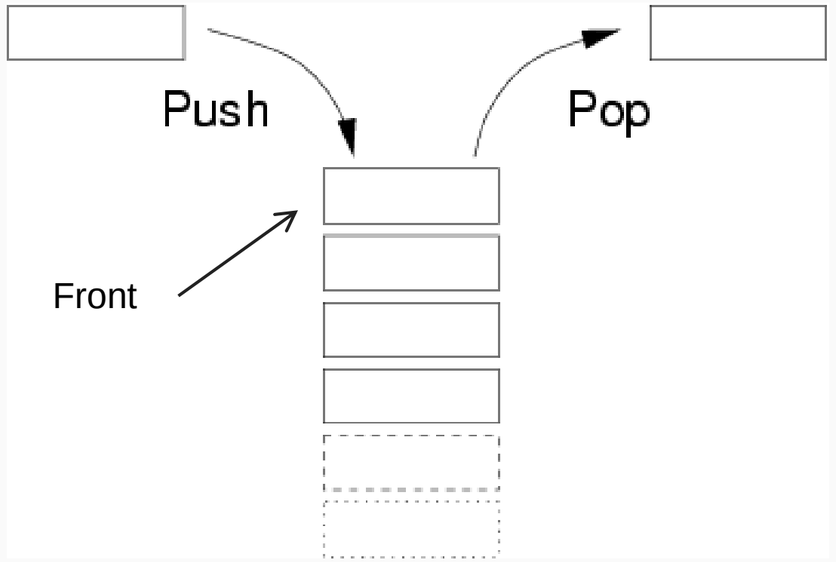
Son estructuras programadas para almacenar datos en memoria (RAM o disco) para que varias operaciones puedan realizarse de manera fácil



# Stack

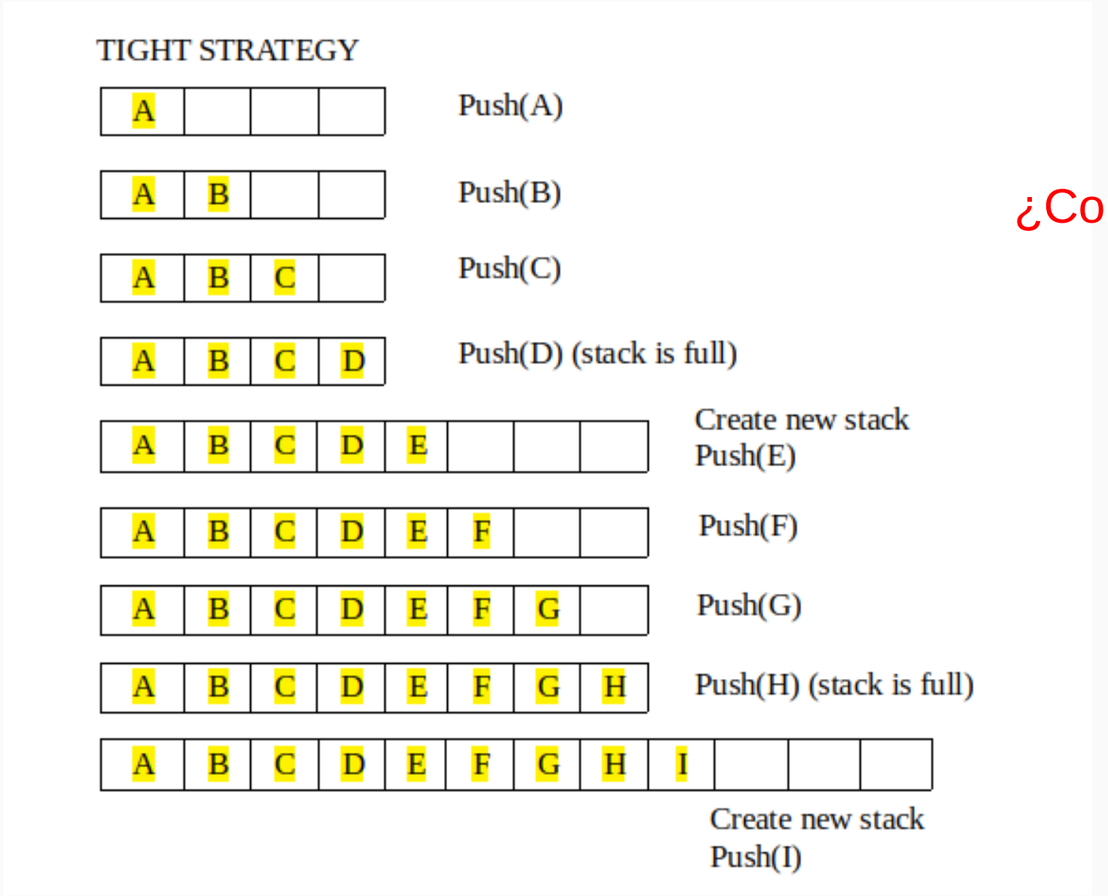
Es una estructura de datos básica que se puede representar como una pila.

Su implementación se llama también LIFO (Last In First Out)



# Stack with Arrays

```
Class Stack {
  Private:
      int* array;
      int capacity;
      int size;
  public:
        Stack();
        void push(int data);
        int pop();
        void display();   }
```

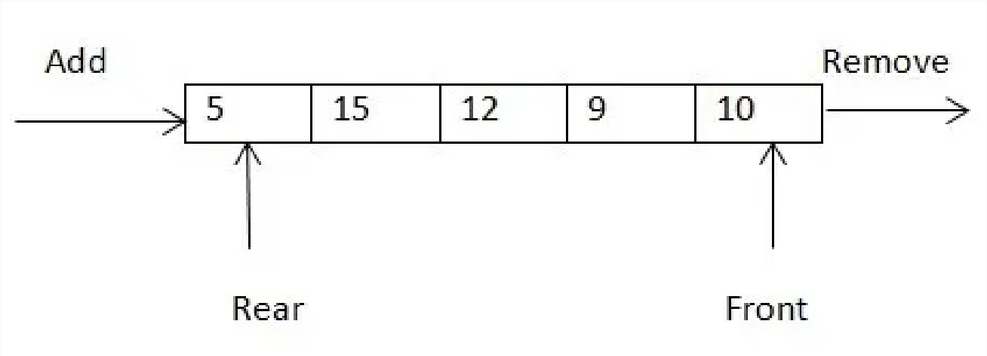



¿Complejidad Computacional?

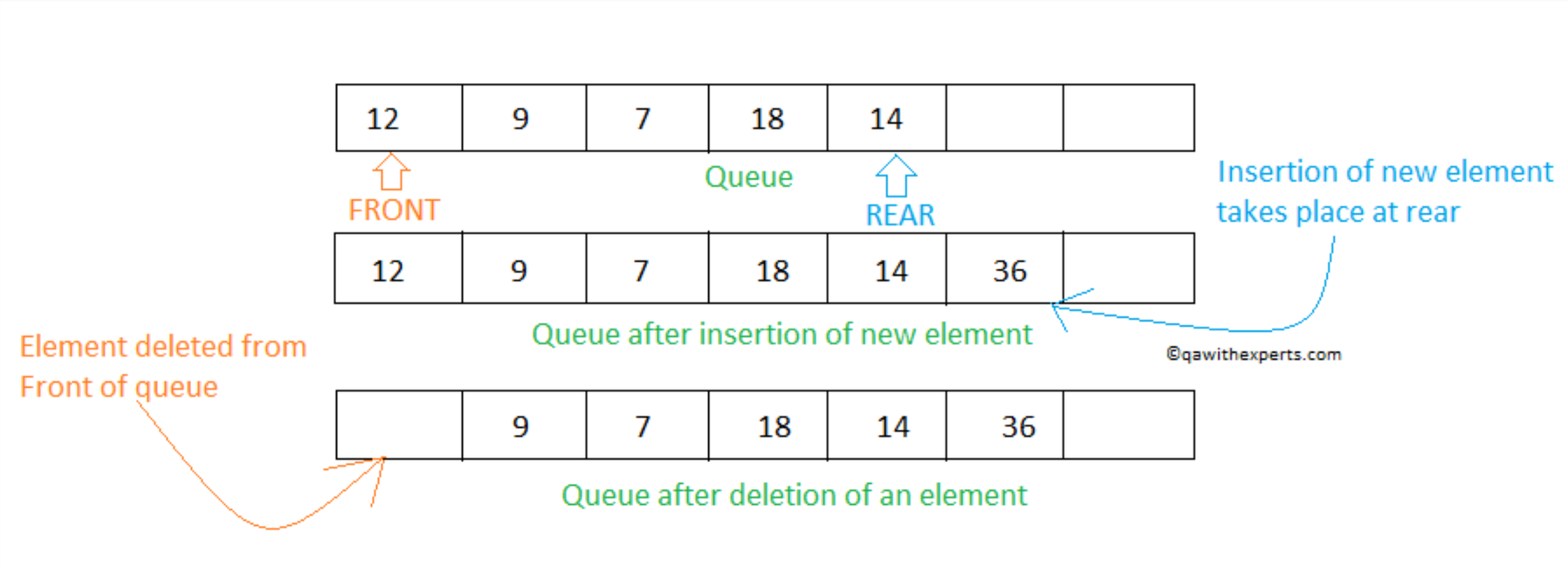
# Queue

Es una estructura de datos básica muy similar a stack.

Su implementación se llama también FIFO (First In First Out)

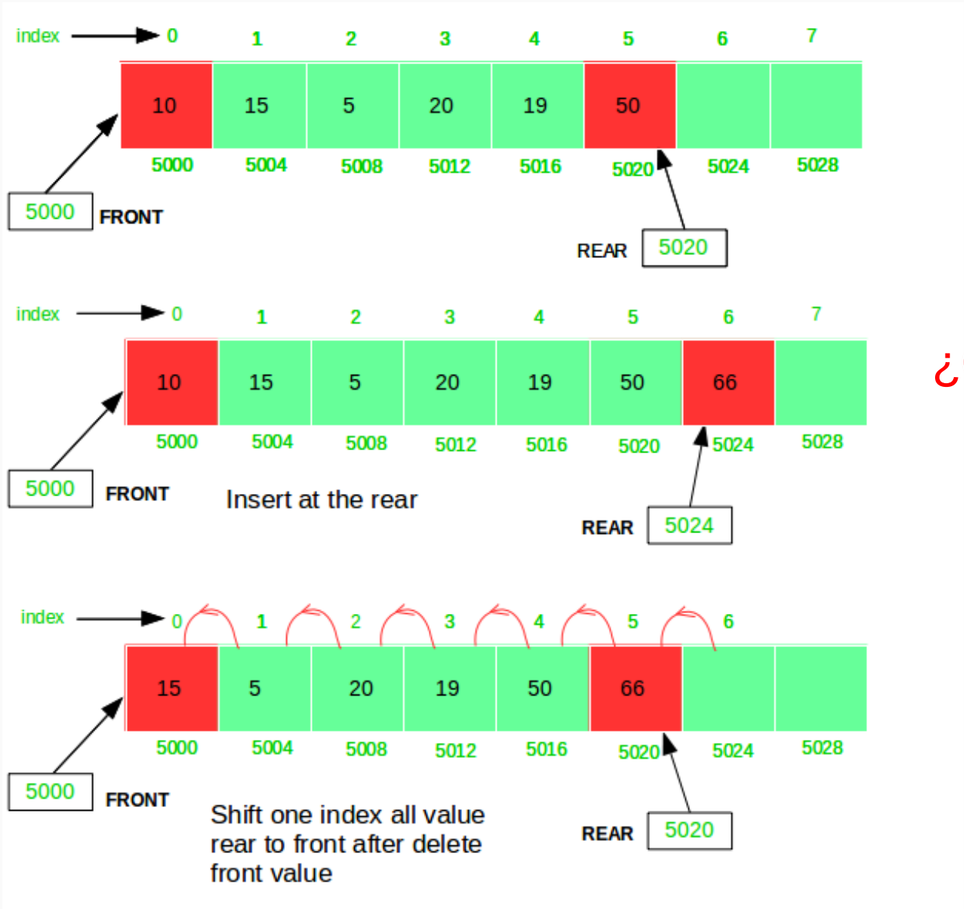


# Queue with Arrays



# Queue with Arrays

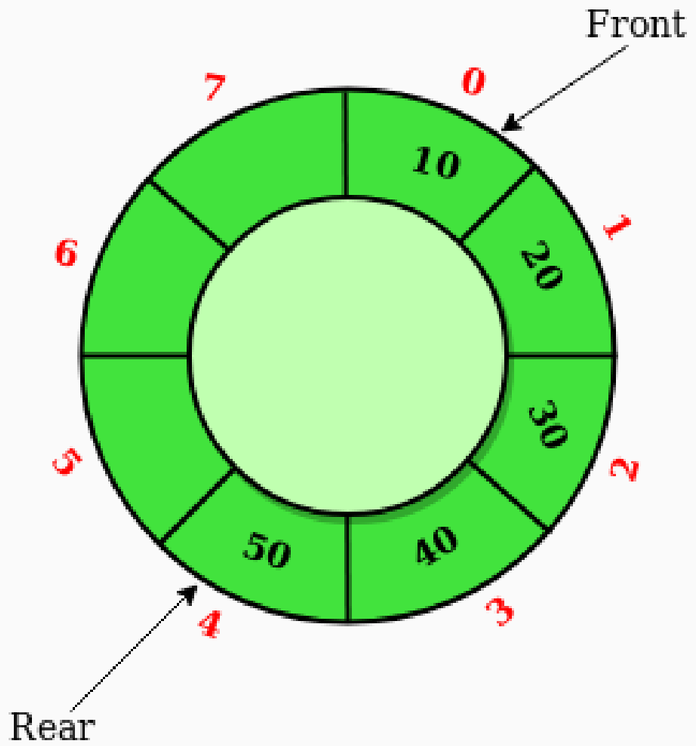
```
Class Queue {
  private:
      int* array;
      int capacity;
      int size;
  public:
        Queue();
        void enqueue(int data);
        int dequeue();
        void display();   }
```




¿Complejidad Computacional?

# Queue with Circular Arrays

```
Class Queue {
  private:
      int* array;
      int capacity;
      int front, rear;
  public:
        Queue();
        void enqueue(int data);
        int dequeue();
        void display();   }
```

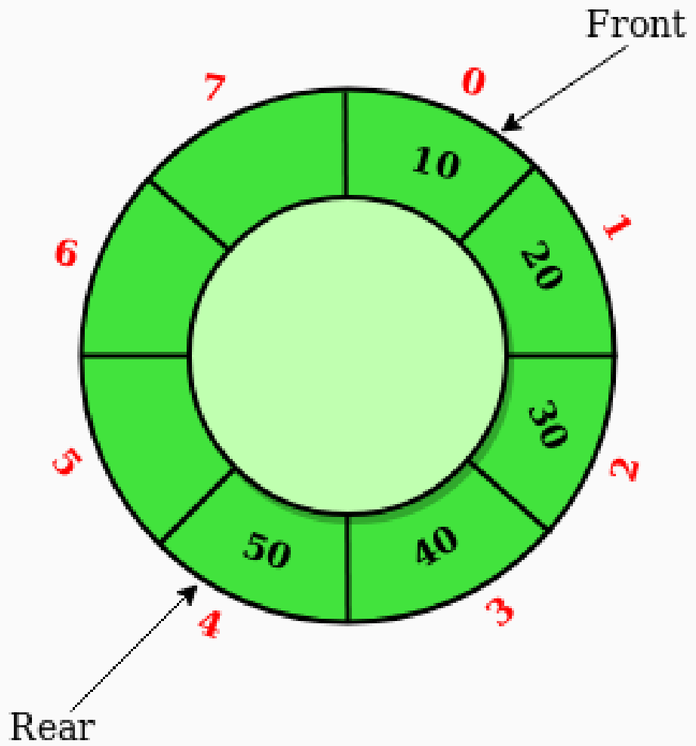



Condiciones para:

- Cola vacía ?
- Cola llena ?

# Queue with Circular Arrays

```
Class Queue {
  private:
      int* array;
      int capacity;
      int front, rear;
  public:
        Queue();
        void enqueue(int data);
        int dequeue();
        void display();   }
```




Condiciones para:

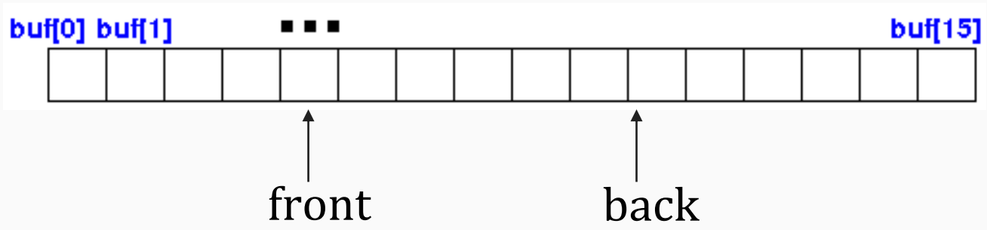
- Cola vacía ?
- Cola llena ?

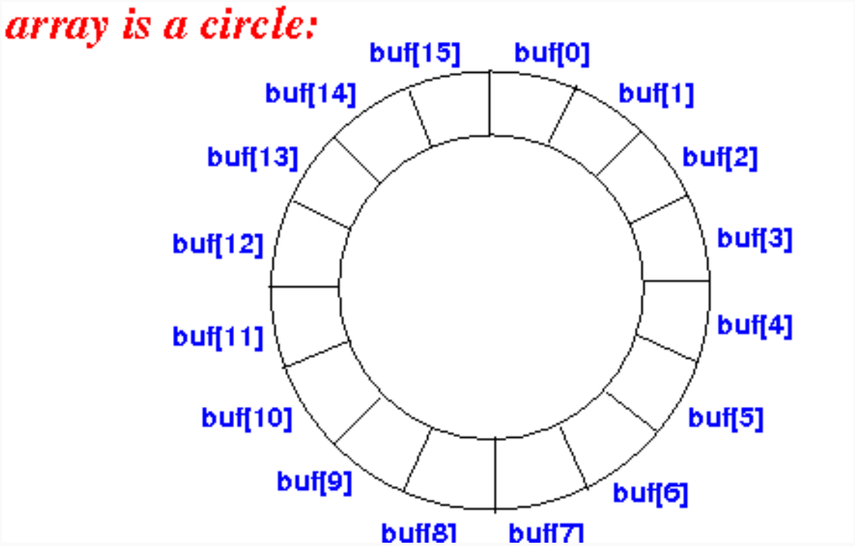
¿Complejidad Computacional?

Insertar: O(1)
Eliminar: O(1)
Problema: redimensionar

# Circular Array

- Operaciones:
  - pop_front()
  - pop_back()
  - push_front(data)
  - push_back(data)
  - **insert(data, position)**
  - Otras:
    - is_empty(), is_full()
- Complejidad:
  - 
  - 
  - 
  - 
  - O(n)



# Circular Array (Homework)

**T front();** *// Retorna el elemento al comienzo*

**T back();** *// Retorna el elemento al final*

**void push_front(T);** *// Agrega un elemento al comienzo*

**void push_back(T);** *// Agrega un elemento al final*

**T pop_front();** *// Remueve el elemento al comienzo*

**T pop_back();** *// Remueve el elemento al final*

**Void insert(T, int);** *// Inserta el elemento en la posición indicada*

**T operator[](int);** *// Retorna el elemento en la posición indicada*

**bool empty();** *// Retorna si el array está vacía o no*

**bool full();** *// Retorna si el array está lleno o no*

**int size();** *// Retorna el tamaño del array*

**void clear();** *// Elimina todos los elementos del array*

**void sort();** *// Ordena el array* ***(use el algoritmo eficiente que le tocó en la práctica de ordenamiento****)*

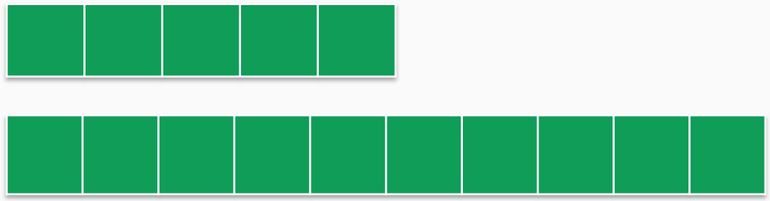
**T* reverse();** // Devuelve una copia d*el array revertido*

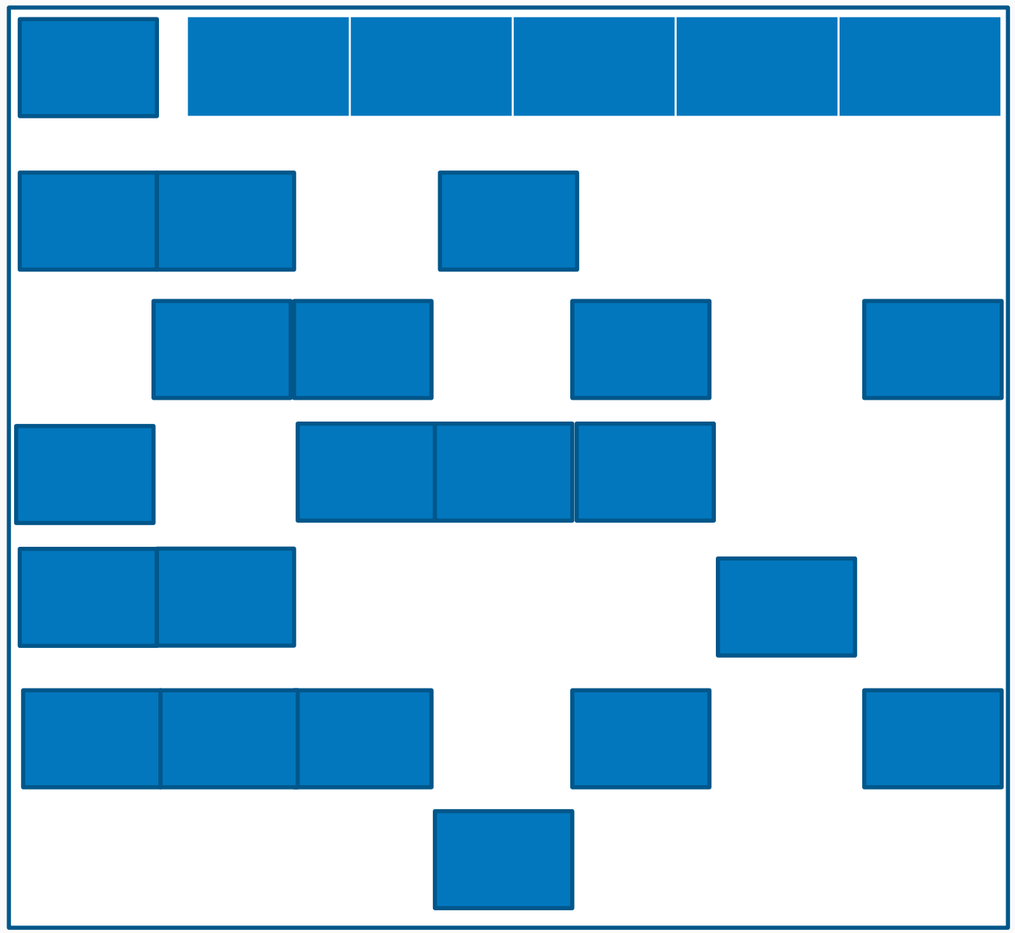
# Array

- Problema de asignación de memoria

**int *array = new int[5]**

**Redimensionar?**





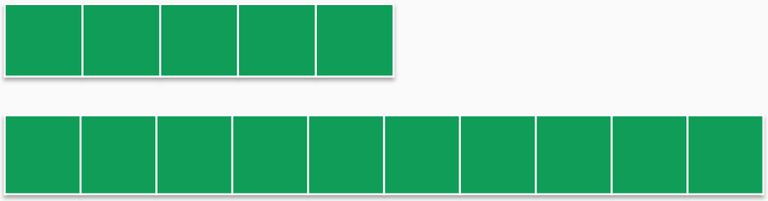
RAM

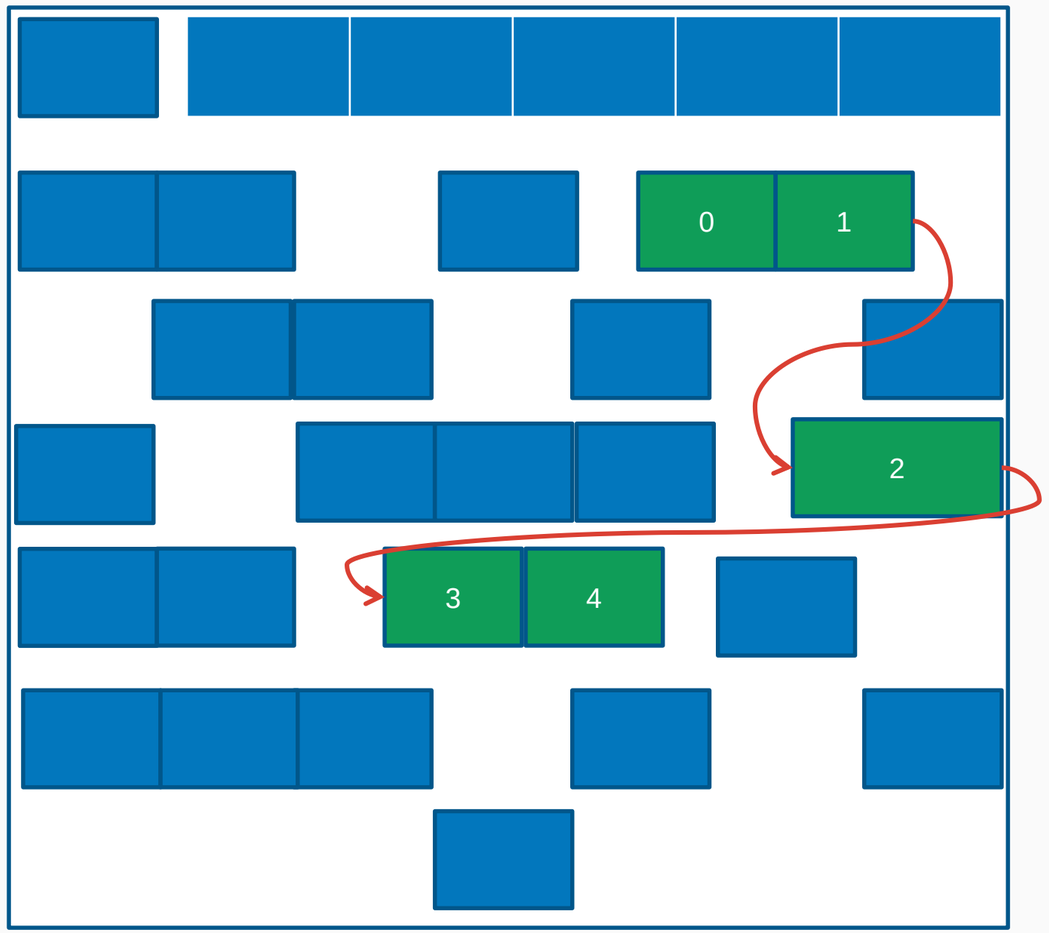
# Array

- Problema de asignación de memoria

**int *array = new int[5]**

**Redimensionar?**






RAM

Solucion: Lista enlazada