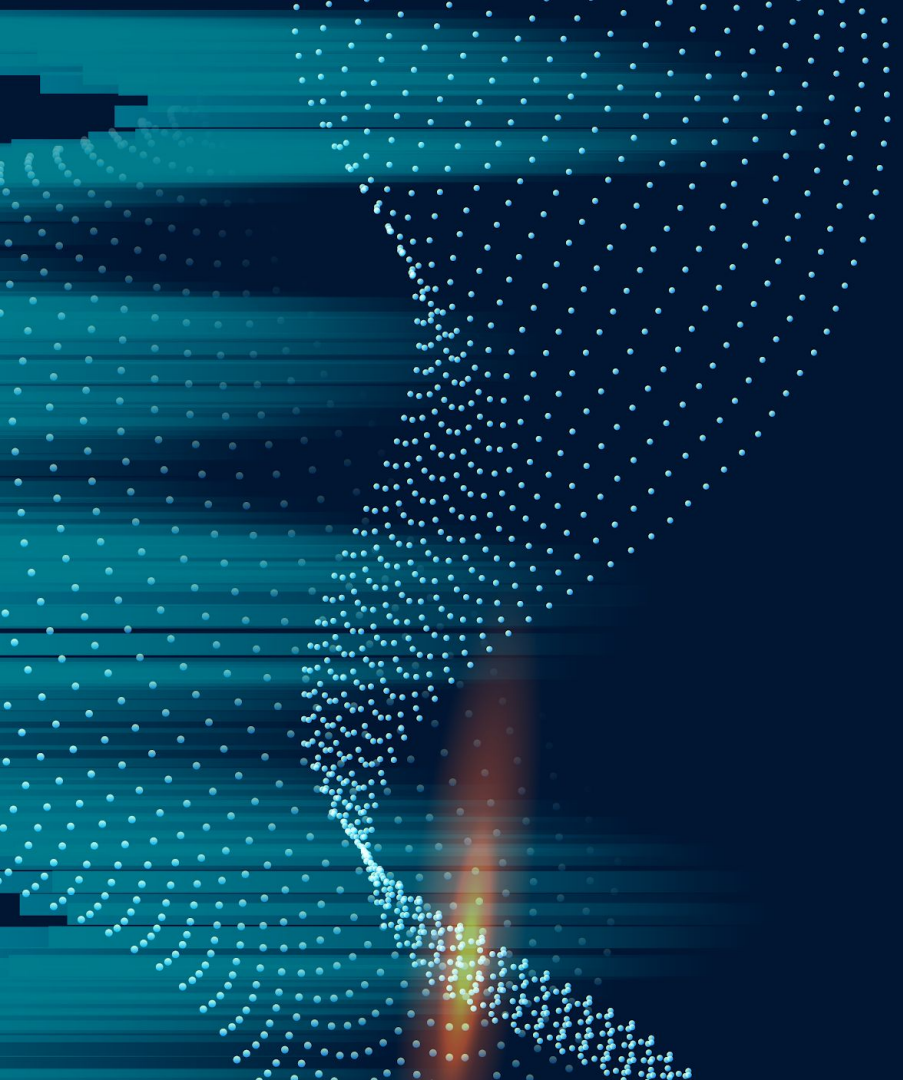
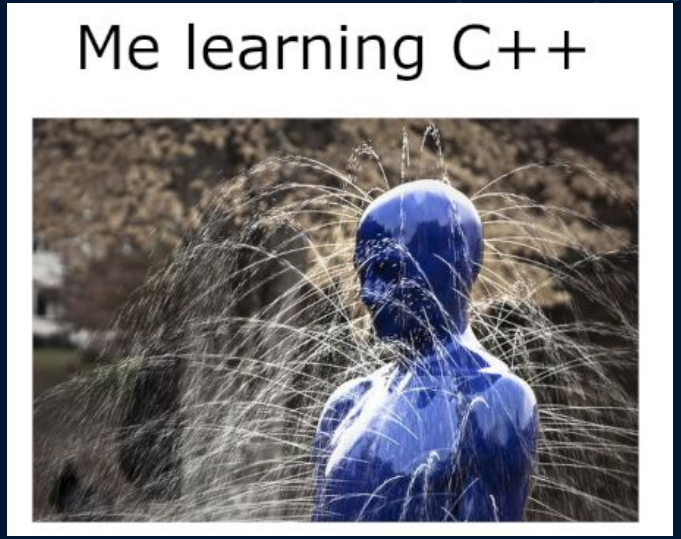
Smart pointers



Memory leak

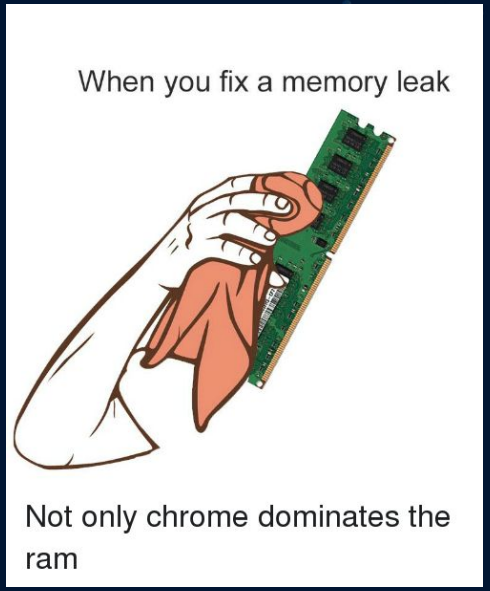
¿Qué es memory leak?

- Es cuando reservas espacio de memoria (*new*) y olvidas liberarlo (*delete*).
- Uno de los errores más comunes es no saber como liberar correctamente la memoria.
- Otro error es olvidar de liberarlo.



¿Por qué es malo?

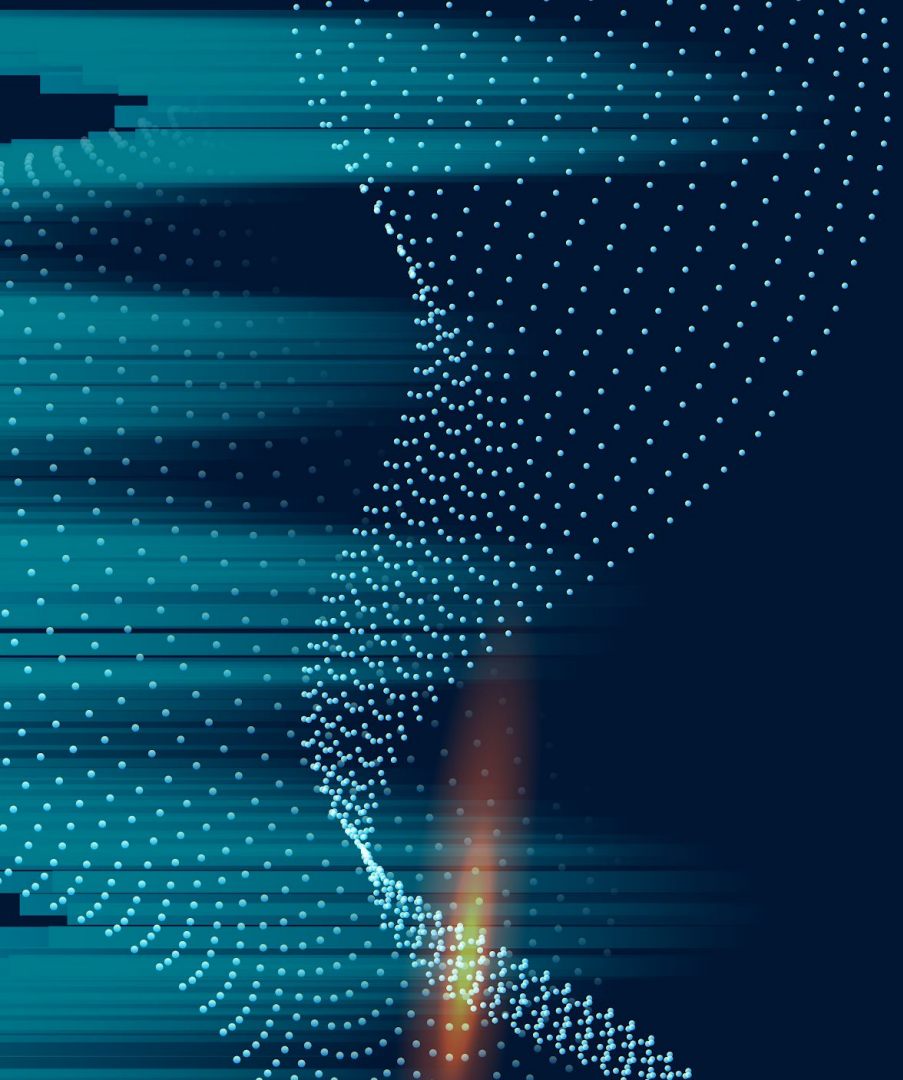
- Todos los sistemas tiene una cantidad limitada de memoria.
- Espacio de memoria desperdiciado.
- Memoria es costosa.



Ejemplo

```cpp
#include <iostream>

using namespace std;

int main() {
  int* puntero = new int(10);

  // falta delete

  return 0;
}
```

```cpp
#include <iostream>

using namespace std;

int main() {
  int *puntero = new int(10);

  delete puntero;

  return 0;
}
```



Smart pointers

¿Qué es un *smart pointer*?

- Es un puntero que libera memoria de manera automática.
- Son más seguros.
- Aparecieron en C++ 11.

¿Cómo se usa?

- Se necesita la librería ***<memory>***.
- Existen 3 tipos de *smart pointers*:
  - **unique_ptr:** puntero con solo un dueño.
  - **shared_ptr:** puntero que puede tener varias referencias al mismo elemento, tiene un contador de referencias.
  - **weak_ptr:** se usa en conjunto con *shared_ptr* pero no participa en el conteo de referencias, sirve para saber si el puntero ha sido liberado o no.

Sintaxis

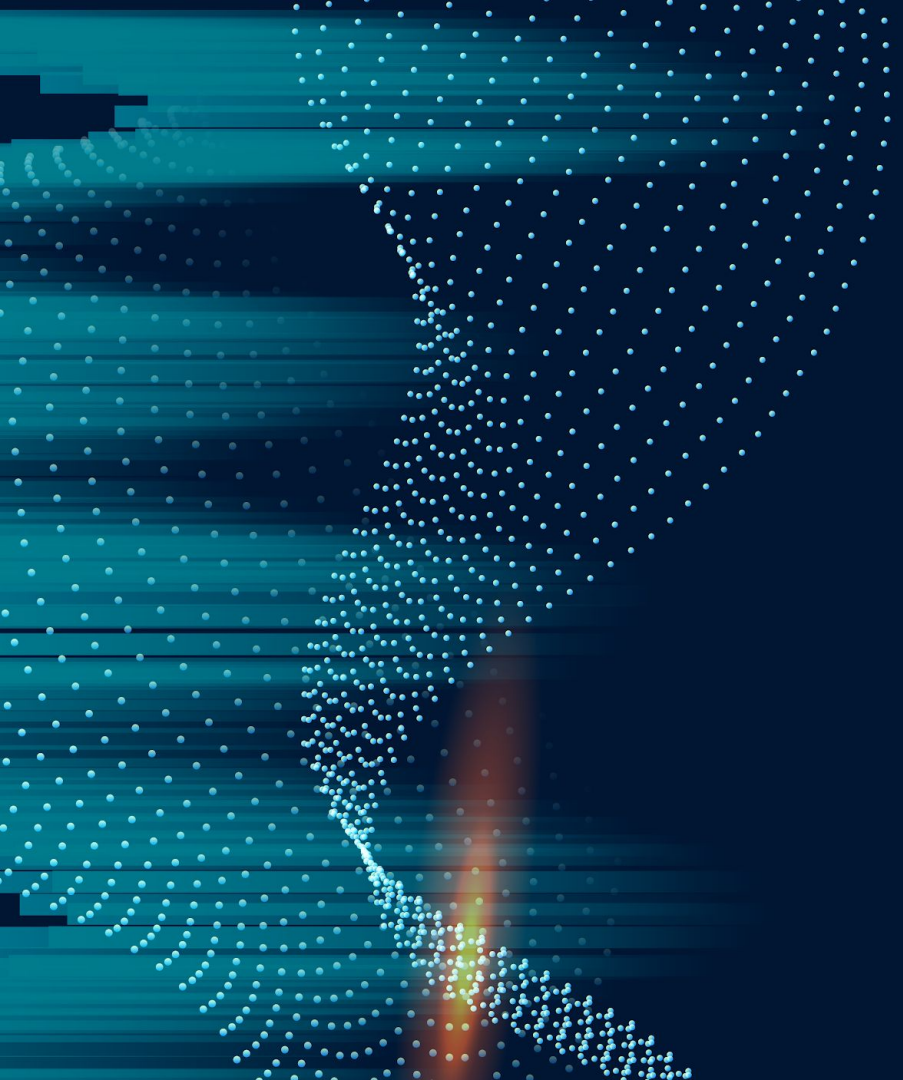
```cpp
#include <iostream>
#include <memory>

using namespace std;

using tipo = int;
tipo dato = 3;

int main() {
  unique_ptr<tipo> ptr1(new tipo(dato));
  unique_ptr<tipo> ptr2 = make_unique<tipo>(dato);

  shared_ptr<tipo> ptr3(new tipo(dato));
  shared_ptr<tipo> ptr4 = make_shared<tipo>(dato);

  weak_ptr<tipo> ptr5(ptr3);
  weak_ptr<tipo> ptr6 = ptr4;

  return 0;
}
```

Métodos útiles

- **get():** retorna la dirección de memoria (*unique* y *shared*).
- **use_count():** retorna la cantidad de referencias que se hacen a un elemento (*shared* y *weak*).
- **reset():** hace *delete* (*unique*, *shared* y *weak*).
- **expire():** retorna *true* si el puntero hizo *delete* y retorna *false* si no (*weak*).
- **lock():** si *expire()* retorna *false* permitirá crear un *shared_ptr*, caso contrario no (*weak*).

Ejemplos:

https://github.com/alonso804/smart-pointers



Raw Pointers vs Smart Pointers

Raw Pointer

- La sintaxis puede ser confusa.
- Riesgo a olvidarse de liberar memoria.
- Se puede usar estáticamente.

Smart Pointer

- La sintaxis es más legible.
- Libera memoria automáticamente.
- En la mayoría de casos no se necesita usar *new*.
- Es más seguro.
- Por defecto es *nullptr*.
- Puede resultar algo verboso.