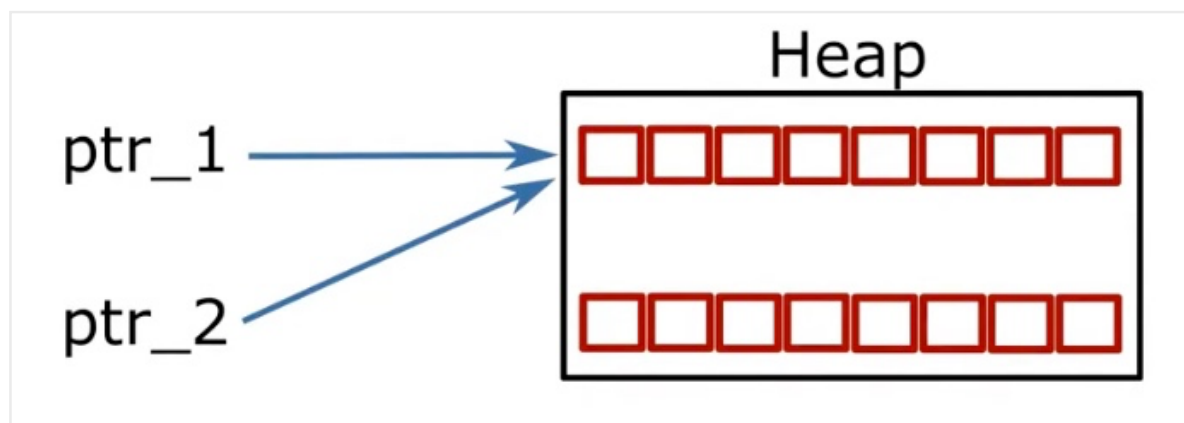
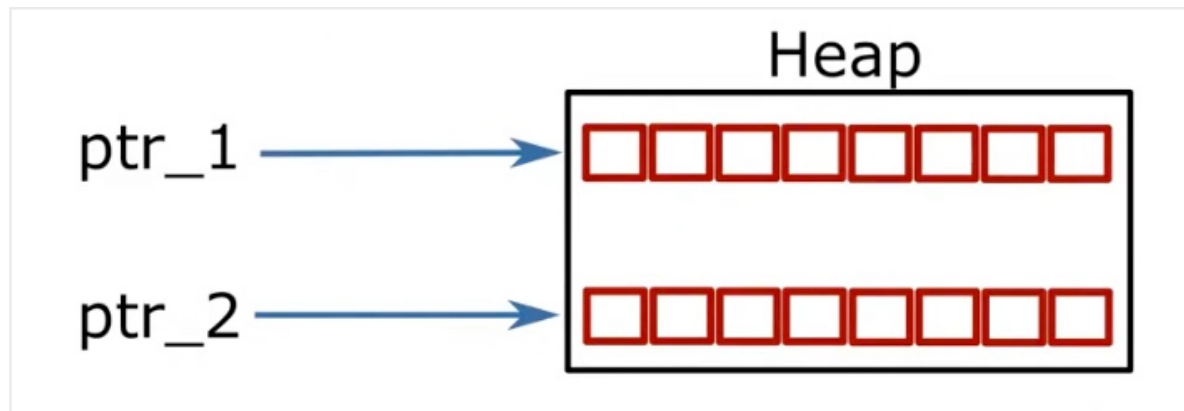


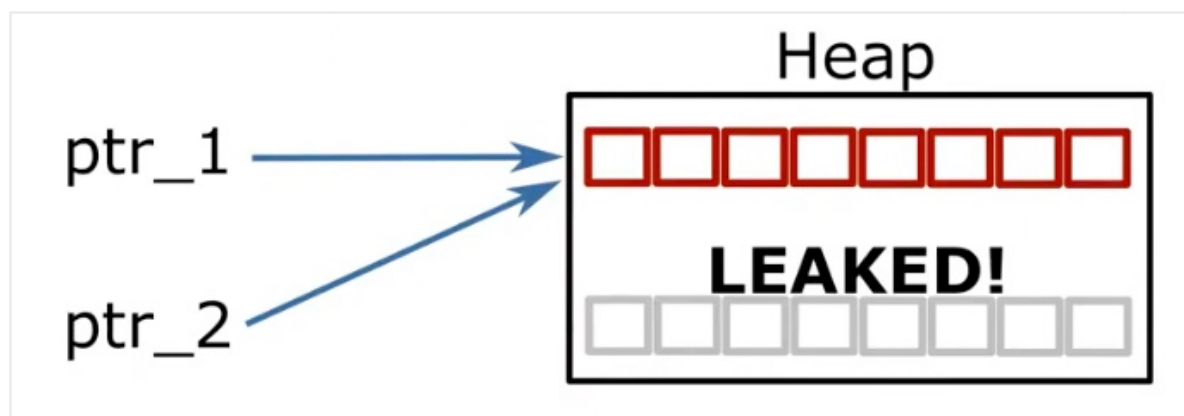
Memory Issues

Memory Leak

- Can happen when working with heap memory if we are not careful
- Memory leak: memory allocated on Heap access to which has been lost



Reference lost!



Memory leak (delete)



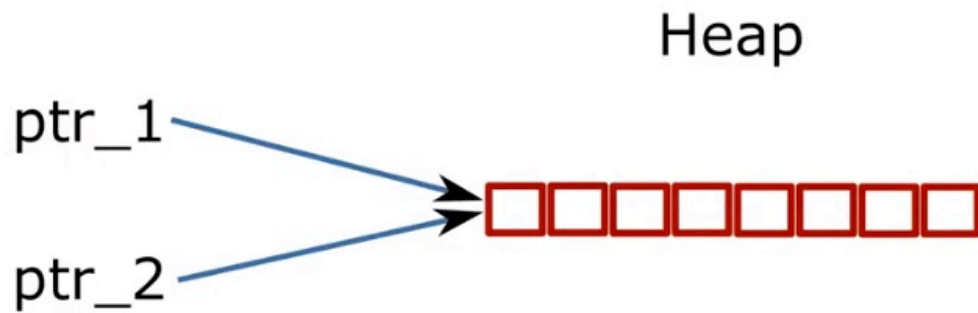
```
1 int main() {
2     int *ptr_1 = nullptr;
3     int *ptr_2 = nullptr;
4
5     // Allocate memory for two bytes on the heap.
6     ptr_1 = new int;
7     ptr_2 = new int;
8     cout << "1: " << ptr_1 << " 2: " << ptr_2 << endl;
9
10    // Overwrite ptr_2 and make it point where ptr_1
11    ptr_2 = ptr_1;
12
13    // ptr_2 overwritten, no chance to access the memory.
14    cout << "1: " << ptr_1 << " 2: " << ptr_2 << endl;
15    delete ptr_1;
16    delete ptr_2;
17    return 0;
18 }
```

Dangling Pointer

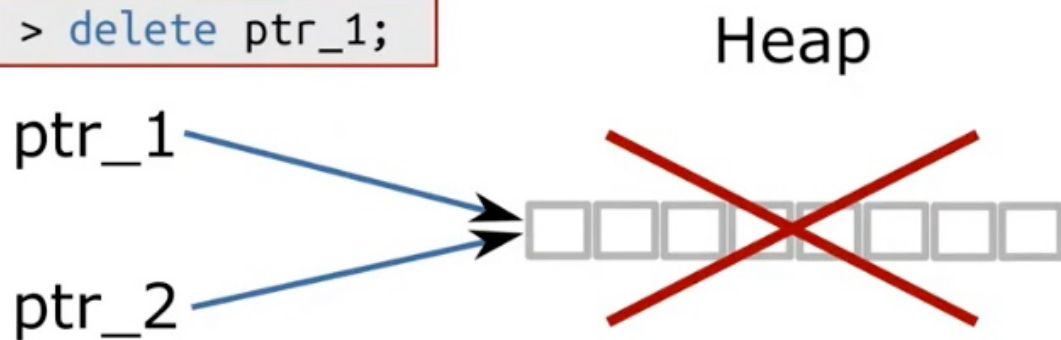
- Dangling Pointer: pointer to a freed memory
- Think of it as the opposite of memory leak
- Dereferencing a dangling pointer causes undefined behaviour

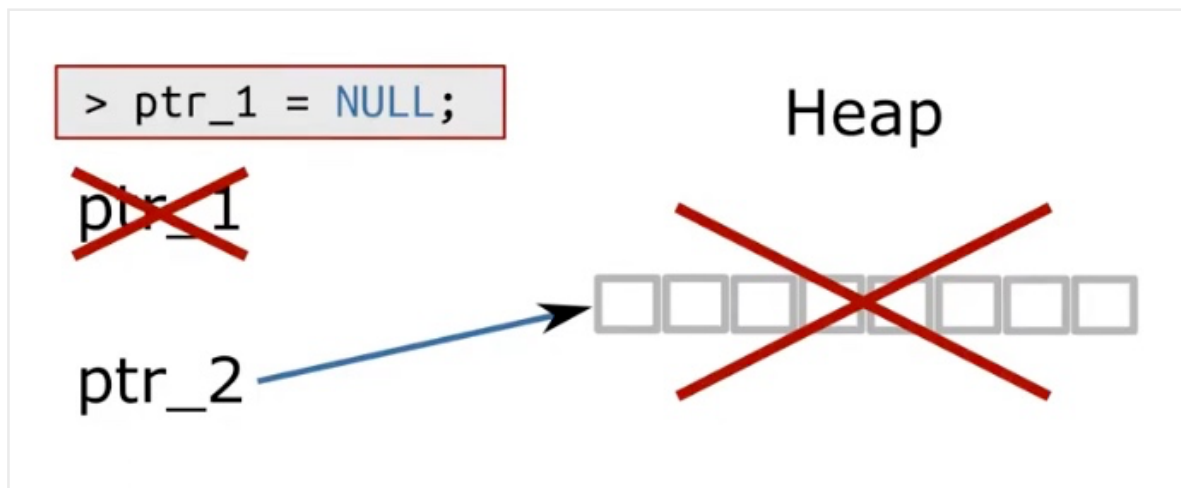
Dangling pointer

```
1 int* ptr_1 = some_heap_address;  
2 int* ptr_2 = some_heap_address;  
3 delete ptr_1;  
4 ptr_1 = nullptr;  
5 // Cannot use ptr_2 anymore! Behavior undefined!
```



```
> delete ptr_1;
```





RAII

- Resource Allocation is Initialization
- New object -> allocate memory
- Remove object -> free memory
- Objects own their data!

```
1 class MyClass {  
2     public:  
3         MyClass() { data_ = new SomeOtherClass; }  
4         ~MyClass() {  
5             delete data_;  
6             data_ = nullptr;  
7         }  
8     private:  
9         SomeOtherClass* data_;  
10 };
```

- Still cannot copy an object of MyClass!!

```

1 struct SomeOtherClass {};
2 class MyClass {
3 public:
4     MyClass() { data_ = new SomeOtherClass; }
5     ~MyClass() {
6         delete data_;
7         data_ = nullptr;
8     }
9 private:
10    SomeOtherClass* data_;
11 };
12 int main() {
13     MyClass a;
14     MyClass b(a);
15     return 0;
16 }

```

*** Error in `raii_example':
double free or corruption: 0x0000000000877c20 ***



Shallow vs Deep copy

- Shallow Copy: just copy pointers, not data
- Deep Copy: copy data, create new pointers
- Default copy constructor and assignment operator implement shallow copying
- RAII + shallow copying -> dangling pointer
- RAII + Rule of All or Nothing (define each special function) -> correct
- Use smart pointers instead!