Subject: Doubt about container
Posted by koldo on Tue, 08 Oct 2019 08:19:31 GMT
View Forum Message <> Reply to Message

I have a doubt about how to keep a class... probably it is extremely simple to solve :)
This is the story:

- A class (A) has to keep a copy of other class (B)
- A can keep:
-- Case 1. a pointer to an instance of B created with new() by A
-- Case 2. a pointer to an instance of B handled in other place
- When A is destroyed
-- Case 1. B is destroyed in A destructor with a delete
-- Case 2. B is not destroyed. The owner has to destroy it

The problem of Case 2 it is that it is possible that B owner destroys B before A is destroyed... so if A is used in between, it can cause problems.

What is the proper way to do it?

Subject: Re: Doubt about container
Posted by mirek on Wed, 09 Oct 2019 09:18:21 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 08 October 2019 10:19I have a doubt about how to keep a class... probably it is extremely simple to solve :)
This is the story:

- A class (A) has to keep a copy of other class (B)
- A can keep:
-- Case 1. a pointer to an instance of B created with new() by A
-- Case 2. a pointer to an instance of B handled in other place
- When A is destroyed
-- Case 1. B is destroyed in A destructor with a delete
-- Case 2. B is not destroyed. The owner has to destroy it

The problem of Case 2 it is that it is possible that B owner destroys B before A is destroyed... so if A is used in between, it can cause problems.

What is the proper way to do it?

It would be more helpful to explain the situation with the code. That said, if the task is to "A class (A) has to keep a copy of other class (B)", then what is wrong with

```
class A {
   B b;
```

}

?

Subject: Re: Doubt about container
Posted by koldo on Thu, 10 Oct 2019 06:49:04 GMT
View Forum Message <> Reply to Message

:lol: :lol:

The examples must be simple, but the simplicity of the examples should not be misunderstood.

As I expected, the answer was terribly easy: Ptr. From Reference/Ptr:

```
struct Foo : Pte<Foo> {String text;};

CONSOLE_APP_MAIN {
 Ptr<Foo> ptr;
 {
  Foo foo;
  foo.text = "Text";
  ptr = &foo;
  Cout() << (void*)~ptr << " -> " << ptr->text << "\n";
 }
 Cout() << (void*)~ptr << "\n"; // foo was destroyed but this is totally valid, as ptr == 0
}
```