Pointer in C.

What is pointer?

A pointer points to an object.

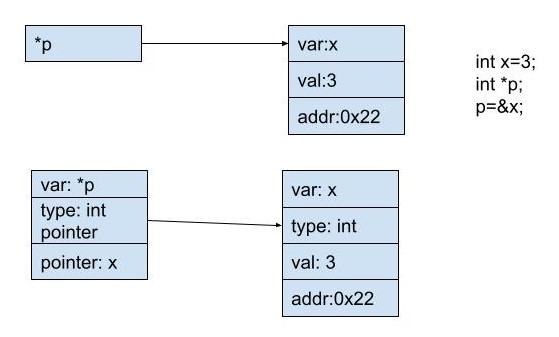
The following figure illustrates how a pointer points to an object.

The following statement

p=&x;

means that the pointer p refers the address of the variable x.

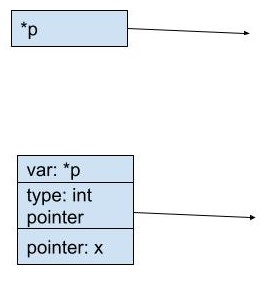
Thus, value (\*p) will be equivalent to value of x.



What is null pointer?

A null pointer points to null (nothing).

Take the following figure for example.



What is nullptr in C++?

nullptr in C++ is a macro which means NULL.

A pointer which is assigned to nullptr is a null pointer.

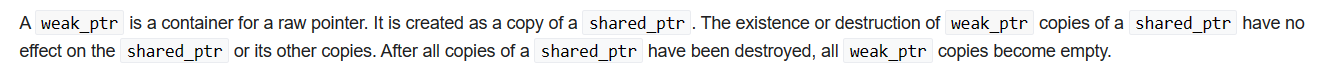
[type]

There are many types of pointer in C and C++.

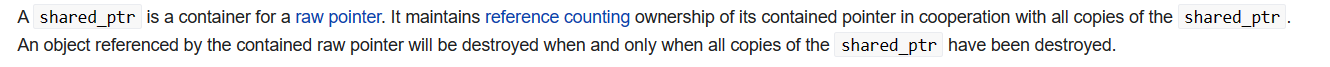
[intro]

Since the auto\_ptr is deprecated, in this section, I will NOT discuss the auto\_ptr.

1. For weak\_ptr:



1. For shared\_ptr:



std::shared\_ptr<int> p0(**new** int(5)); *// Valid, allocates 1 integer and initialize it with value 5.*

std::shared\_ptr<int[]> p1(**new** int[5]); *// Valid, allocates 5 integers.*

std::shared\_ptr<int[]> p2 = p1; *// Both now own the memory.*

p1.reset(); *// Memory still exists, due to p2.*

p2.reset(); *// Frees the memory, since no one else owns the memory.*

std::shared\_ptr<int> p1 = std::make\_shared<int>(5);

std::weak\_ptr<int> wp1 {p1}; *// p1 owns the memory.*

{

std::shared\_ptr<int> p2 = wp1.lock(); *// Now p1 and p2 own the memory.*

*// p2 is initialized from a weak pointer, so you have to check if the*

*// memory still exists!*

**if** (p2) {

DoSomethingWith(p2);

}

}

*// p2 is destroyed. Memory is owned by p1.*

p1.reset(); *// Free the memory.*

std::shared\_ptr<int> p3 = wp1.lock();

*// Memory is gone, so we get an empty shared\_ptr.*

**if** (p3) { *// code will not execute*

ActionThatNeedsALivePointer(p3);

}

(3)For unique\_ptr

std::unique\_ptr<int> p1(**new** int(5));

//std::unique\_ptr<int> p2 = p1; *// Compile error.*

std::unique\_ptr<int> p3 = std::move(p1); *// Transfers ownership. p3 now owns the memory and p1 is set to nullptr.*

p3.reset(); *// Deletes the memory.*

p1.reset(); *// Does nothing.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| item | stored pointer | | store deleter | |
| action | What situation when value is changed? | Value can be accessed via? | What situation when value is changed? | Value can be accessed via? |
| shared\_ptr | One the following cases.  (1)Assignment.  (2)Calling method reset. | Method of get | One the following cases.  (1)Assignment.  (2)Calling method reset. | Method of get\_deleter |
| Unique\_ptr | Method of get and release |

|  |  |  |
| --- | --- | --- |
| Item |  |  |
| Garbage collection facility | Yes (But limited) | Yes (But limited) |
| Objects ability of taking ownership | Yes | Yes |
| Delete the management when | (1)They are destroyed.  (2)Value changed either by assignment.  (3)Value changed either by explicit call of reset. | (1)They are destroyed.  (2)Value changed either by assignment.  (3)Value changed either by explicit call of reset. |
| Ownership relationship  (Can they be shared?) | They can be shared. | Unique, they can NOT be shared.  Unique\_ptr own the pointer unique.  i.e. No other facility should take care of deleting the object. |

[ref]

1. Introduction to weak\_ptr, shared\_ptr, unique\_ptr on the following link.

<https://en.wikipedia.org/wiki/Smart_pointer>

1. For method and properties etc on the following links.

<https://cplusplus.com/reference/memory/weak_ptr/weak_ptr/>

<https://cplusplus.com/reference/memory/shared_ptr/>

<https://cplusplus.com/reference/memory/unique_ptr/>

<https://cplusplus.com/reference/memory/>