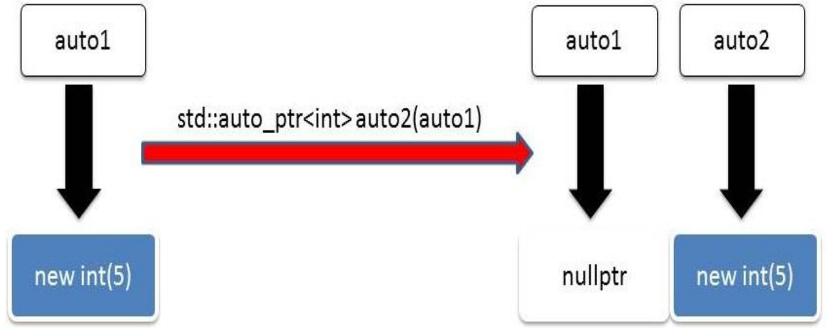
std::auto_ptr versus std::unique_ptr

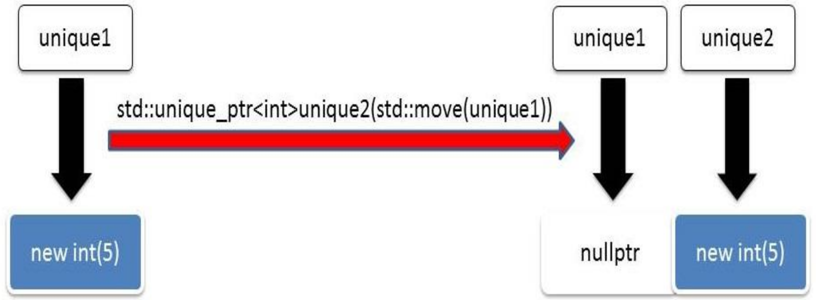
std::auto_ptr

```
std::auto_ptr<int> auto1(new int(5));
std::auto_prt<int> auto2(auto1);
```



std::unique_ptr

```
std::unique_ptr<int> unique1(new int(5));
std::unique_ptr<int> unique2(std::move(unique1));
```



`std::unique_ptr`

The `std::unique_ptr` automatically manages the lifetime of its resource, following the RAII-Idiom.

- `std::unique_ptr`
  - is the replacement for the *deprecated* `std::auto_ptr`.

    ➡ `std::unique_ptr` can not be copied.
  - can be used in the STL container.

    ➡ container must not have copy semantic.
  - has *minimal* management overhead.
  - can be parametrized by a delete: `std::unique_ptr<T, Deleter>`.
  - is specialized for C arrays: `std::unique_ptr<T[]>`.

std::unique_ptr

Member Function	Description
`uniq.release()`	Returns a pointer to the resource and release it.
`uniq.get()`	Returns a pointer to the resource.
`uniq.reset(ptr)`	▪ Replaces the resource. ▪ Destructs the old resource.
`uniq.swap(uniq2)`	Swaps to `std::unique_ptr`.
`uniq.get_deleter()`	Returns the delete function.
`std::make_unique(...)`	▪ Generates the resource and returns a `std::unique_ptr`. ▪ Is available since C++14.

uniquePtr.cpp