Умные указатели в C++

Умные указатели (smart pointers) — это механизмы управления памятью в языке C++, которые автоматически следят за временем жизни объектов, освобождая память при необходимости. Они реализованы в библиотеке `<memory>` и помогают избежать утечек памяти, ошибок двойного освобождения, а также упрощают управление ресурсами. Рассмотрим основные виды умных указателей:

1 scoped_ptr

`scoped_ptr` был частью библиотеки Boost, но не вошел в стандарт C++ и в современных версиях заменен на `unique_ptr`. Он представляет собой простой умный указатель с ограниченной областью видимости. Как только объект выходит из области видимости, его память автоматически освобождается.

Основные свойства:

- Не позволяет передавать владение (копировать или перемещать указатель).
- Всегда освобождает объект при выходе из области видимости.
- Подходит для управления объектами с жесткой локальной областью действия.

Пример:

```
#include <boost/scoped_ptr.hpp>
#include <iostream>

void example() {
    boost::scoped_ptr<int> ptr(new int(42));
    std::cout << *ptr << std::endl;
    // ptr
}
```

Недостаток:

- Ограниченная функциональность. В стандарте C++11 и позже `unique_ptr` выполняет аналогичную роль.

2 unique_ptr

`unique_ptr` — это умный указатель, который представляет собой эксклюзивное владение объектом. Это значит, что в каждый момент времени только один `unique_ptr` может владеть объектом.

Основные свойства:

- **Эксклюзивное владение:** объект принадлежит только одному указателю.
- **Перемещение:** владение можно передать другому `unique_ptr` с помощью механизма перемещения (`std::move`).
- **Автоматическое освобождение:** освобождает ресурс при выходе из области видимости.

Пример:

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

void example() {
    std::unique_ptr<int> ptr1(new int(42));
    std::cout << *ptr1 << std::endl;

    std::unique_ptr<int> ptr2 = std::move(ptr1); //
    if (!ptr1) {
        std::cout << "ptr1                ." << std::endl;
    }
    std::cout << *ptr2 << std::endl;
    // ptr2
}
```

Особенности:

- Нельзя копировать (`unique_ptr<int> ptr2 = ptr1;` вызовет ошибку).
- Можно использовать пользовательские функции удаления через `unique_ptr<T, Deleter>`.

3 shared_ptr

`shared_ptr` — это умный указатель, позволяющий разделять владение объектом между несколькими `shared_ptr`. Объект будет удален только тогда, когда последний `shared_ptr`, владеющий им, выйдет из области видимости.

Основные свойства:

- **Разделяемое владение:** несколько `shared_ptr` могут владеть одним объектом.
- **Счетчик ссылок:** автоматически увеличивает/уменьшает счетчик ссылок на объект.
- **Автоматическое удаление:** объект удаляется, когда счетчик ссылок достигает нуля.

Пример:

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

void example() {
    std::shared_ptr<int> ptr1 = std::make_shared<int>(42); //        make_shared
    std::cout << *ptr1 << std::endl;

    std::shared_ptr<int> ptr2 = ptr1; //
    std::cout << "Count: " << ptr1.use_count() << std::endl; //

    ptr1.reset(); // ptr1
    std::cout << "Count: " << ptr2.use_count() << std::endl; // ptr2
}
```

Особенности:

- Более затратный по сравнению с `unique_ptr` из-за управления счетчиком ссылок.
- Не подходит для циклических ссылок (решается с помощью `weak_ptr`).

4 Сравнение

Указатель	Ключевая особенность	Передача владения	Счетчик ссылок	Применение
`scoped_ptr`	Простота, локальная область	Нет	Нет	Устарел, заменен на
`unique_ptr`	Эксклюзивное владение	Да (через move)	Нет	Управление уникальн
`shared_ptr`	Разделяемое владение	Да	Да	Совместное использо

Итог

- Используйте `unique_ptr` для объектов с уникальным владением.

- Используйте `shared_ptr` для объектов, которыми должны владеть несколько частей программы.
- Избегайте устаревшего `scoped_ptr`, заменяя его на современные аналоги.