Умные указатели:

unique\_ptr, функция make\_unique

Стандартная библиотека C++ включает смарт-указатели, которые используются для обеспечения того, чтобы программы были свободны от утечки памяти (неконтролируемое уменьшение свободной оперативной или виртуальной памяти компьютера) и ресурсов и являются исключениями.

В отличие от управляемых языков, C++ не имеет автоматической сборки мусора, внутренний процесс, который освобождает кучи памяти и другие ресурсы в процессе выполнения программы. Программа C++ отвечает за возврат всех приобретенных ресурсов в операционную систему. Неиспользуемый ресурс называется утечкой. Утечка ресурсов недоступна другим программам до тех пор, пока процесс не завершится. Утечки памяти, в частности, являются распространенными причинами ошибок в программировании в стиле C.

В следующем примере сравниваются объявления необработанного и интеллектуального указателей:

void UseRawPointer() {

// Using a raw pointer -- not recommended.

Song\* pSong = new Song(L"Nothing on You", L"Bruno Mars");

// Use pSong...

// Don't forget to delete!

delete pSong;

}

void UseSmartPointer() {

// Declare a smart pointer on stack and pass it the raw pointer.

unique\_ptr<Song> song2(new Song(L"Nothing on You", L"Bruno Mars"));

// Use song2...

wstring s = song2->duration\_;

//...

} // song2 is deleted automatically here.

Как показано в примере, интеллектуальный указатель — это шаблон класса, который объявляется в стеке и инициализируется с помощью необработанного указателя, указывающего на размещенный в куче объект. После инициализации интеллектуальный указатель становится владельцем необработанного указателя. Это означает, что интеллектуальный указатель отвечает за удаление памяти, заданной необработанным указателем. Деструктор интеллектуального указателя содержит вызов для удаления, и поскольку интеллектуальный указатель объявлен в стеке, его деструктор вызывается, как только интеллектуальный указатель оказывается вне области, даже если исключение создается где-либо в другой части стека.

Интеллектуальные указатели определяются в std пространстве имен в файле заголовка <memory>.

Указатель unique\_ptr<T> представляет указатель на тип T, который является "уникальным" в том смысле, что что может быть только один объект unique\_ptr, который содержит один и тот же адрес. То есть не может одновременно быть двух или более объектов unique\_ptr<T>, которые указывают один и тот же адрес памяти. Если же мы попробуем определить два одновременно существующих указателя, которые указывают на один и тот же адрес, компилятор не скопирует код.

И когда unique\_ptr уничтожается, уничтожается и значение, на которое он указывает. Соответственно данный тип указателей полезен, когда нужен указатель на объект, на который НЕ будет других указателей и который будет удален после удаления указателя.

По умолчанию unique\_ptr<T> инициализируется значением nullptr.

std::unique\_ptr<int> ptr; // ptr = nullptr

// аналогично

std::unique\_ptr<int> ptr{};

std::unique\_ptr<int> ptr{nullptr};

Чтобы выделить память и создать в ней объект, на который будет указывать указатель, применяется функция std::make\_unique<T>. В качестве параметра в нее передается объект, на который будет указывать указатель:

std::unique\_ptr<int> ptr { std::make\_unique<int>(125) };

Стоит отметить, что до принятия стандарта C++14 применялась другая форма создания указателя:

std::unique\_ptr<int> ptr { new int(125) };

Для получения стандартного указателя из std::unique\_ptr применяется функция get():

std::unique\_ptr<int> ptr { std::make\_unique<int>(125) };

int\* pointer = ptr.get();

Стоит отметить, что начиная со стандарта C++ 20 получить адрес из smart-указателя мы можем напрямую без функции get.

После определения интеллектуального указателя мы можем получать и изменять значение, на которое он указывает, так же, как и при работе с обычными указателями:

// указатель ptr указывает на объект 125

std::unique\_ptr<int> ptr { std::make\_unique<int>(125) };

std::cout << "Address: " << ptr.get() << std::endl; // получим адрес объекта

std::cout << "Initial value: " << \*ptr << std::endl; // получим значение объекта

// изменяем значение

\*ptr = 254;

std::cout << "New value: " << \*ptr << std::endl; // получим значение объекта

Если потребуется освободить память, на которую указывает указатель, то можно применить функцию reset().

Источники:

* <https://learn.microsoft.com/ru-ru/cpp/cpp/smart-pointers-modern-cpp?view=msvc-170>
* <https://learn.microsoft.com/ru-ru/cpp/standard-library/memory?view=msvc-170>
* <https://metanit.com/cpp/tutorial/11.1.php>

Подготовил Занин Вячеслав ВМК-21