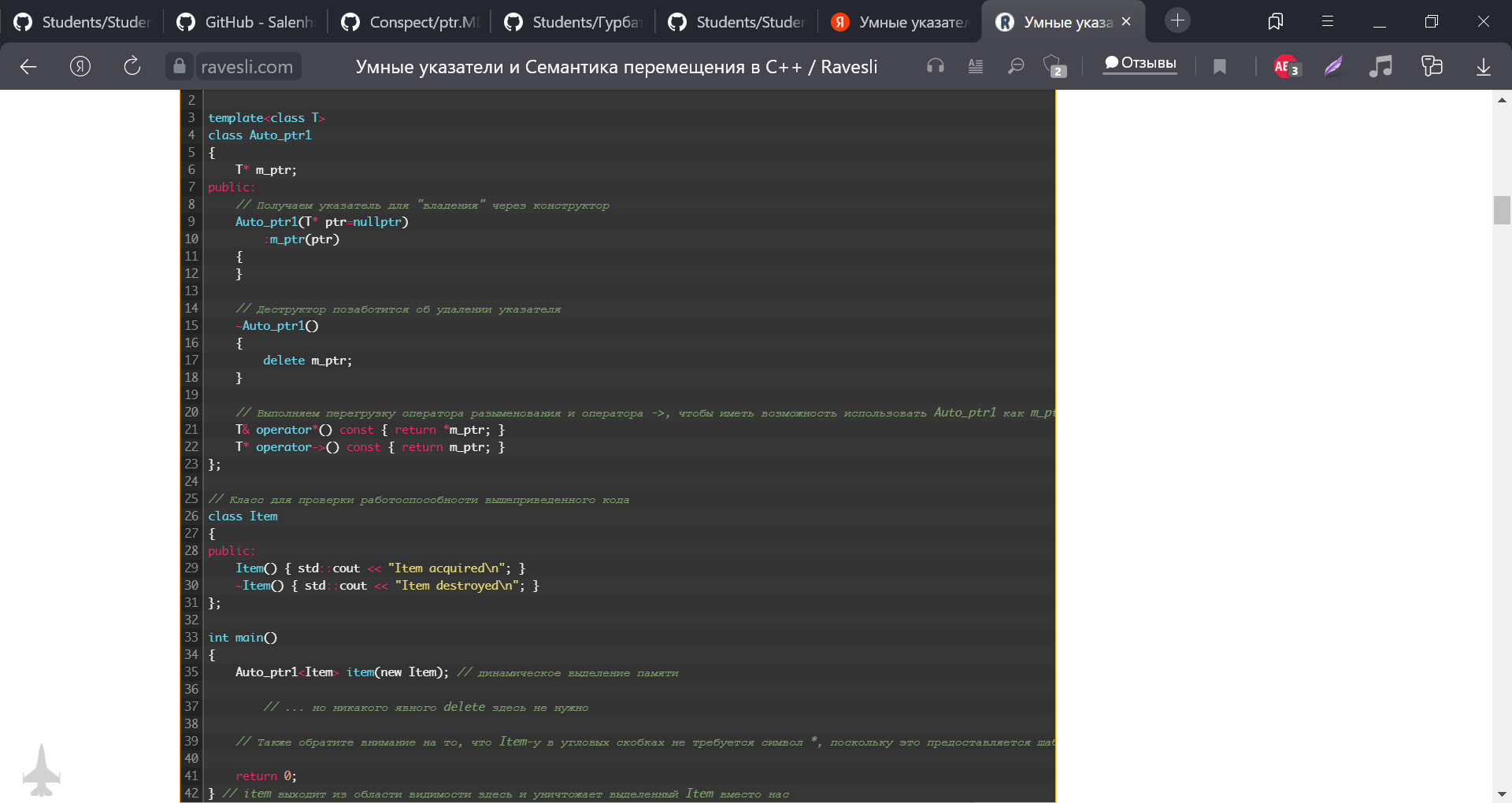
**Умный указатель** — это класс, предназначенный для управления динамически выделенной памятью и обеспечения освобождения (удаления) выделенной памяти при выходе объекта этого класса из области видимости.

Одна из лучших особенностей [**классов**](https://ravesli.com/urok-113-klassy-obekty-i-metody-klassov/) — это [**деструкторы**](https://ravesli.com/urok-120-destruktory/), которые автоматически выполняются при выходе объекта класса из области видимости. При выделении памяти в [**конструкторе**](https://ravesli.com/urok-116-konstruktory/) класса, вы можете быть уверены, что эта память будет освобождена в деструкторе при уничтожении объекта класса (независимо от того, выйдет ли он из области видимости, будет ли явно удален и т.д.).

Например, рассмотрим класс, единственными задачами которого является хранение и «управление» переданным ему указателем, а затем корректное освобождение памяти при выходе объекта класса из области видимости.



Сначала мы динамически выделяем объект класса Item и передаем его в качестве параметра нашему [**шаблону класса**](https://ravesli.com/urok-175-shablony-klassov/) Auto\_ptr1. С этого момента объект item класса Auto\_ptr1 владеет выделенным объектом класса Item (Auto\_ptr1 имеет [**композиционную связь**](https://ravesli.com/urok-147-kompozitsiya-obektov/) с m\_ptr). Поскольку item объявлен в качестве локальной переменной и имеет область видимости блока, он выйдет из области видимости после завершения выполнения блока, в котором находится, и будет уничтожен. А поскольку это объект класса, то при его уничтожении будет вызван деструктор Auto\_ptr1. Этот деструктор и обеспечит удаление указателя Item, который он хранит.

Класс Auto\_ptr1, приведенный выше, имеет критическую ошибку, которая скрывается за некоторым автоматически генерируемым кодом.

Поскольку мы не предоставили [**конструктор копирования**](https://ravesli.com/urok-141-konstruktor-kopirovaniya/) или свой оператор присваивания (перегрузку оператора присваивания), то язык C++ предоставил их самостоятельно. И то, что он предоставил, выполняет [**поверхностное копирование**](https://ravesli.com/urok-145-poverhnostnoe-i-glubokoe-kopirovanie/#toc-0).

Мы можем явно определить и удалить конструктор копирования с оператором присваивания, тем самым предотвращая выполнение любого копирования. Это также предотвратит передачу по значению.

**Семантика перемещения** означает, что класс, вместо копирования, передает право собственности на объект.

## !!!НЕ ИСПОЛЬЗУЙТЕ std::auto\_ptr!!!

Вместо него используйте **другие типы умных указателей:**

   std::scoped\_ptr;

   std::unique\_ptr;

   std::weak\_ptr;

   std::shared\_ptr.

<https://ravesli.com/urok-189-umnye-ukazateli-i-semantika-peremeshheniya/>