***Умные указатели.***

Умный указатель — это класс для управления памятью. Экземпляр этого класса создаётся статически. Он сам позаботится о том, чтобы освободить память другого объекта, созданного динамически. Инициализировать умный указатель нужно сразу.

Встроенные (обычные) указатели иногда еще называют «глупыми указателями», так как они не могут выполнять после себя очистку памяти.

Как правило, основной целью задействования умных указателей является [инкапсуляция](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) работы с [динамической памятью](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8) таким образом, чтобы свойства и поведение умных указателей имитировали свойства и поведение обычных указателей. При этом на них возлагается обязанность своевременного и аккуратного высвобождения выделенных ресурсов, что упрощает разработку кода и процесс отладки, исключая [утечки памяти](https://ru.wikipedia.org/wiki/%D0%A3%D1%82%D0%B5%D1%87%D0%BA%D0%B0_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8) и возникновение [висячих ссылок](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%81%D1%8F%D1%87%D0%B0%D1%8F_%D1%81%D1%81%D1%8B%D0%BB%D0%BA%D0%B0).

* [Обращение к ресурсу происходит в один этап. Либо мы получаем готовый полностью функциональный объект сразу, либо не получаем ничего.](http://virtser.net/blog/post/source-code-highlighter.aspx)
* [Безопасность по отношению к исключению. Например, если после создания объекта и обращения к ресурсу произойдет исключение, и мы перейдем к обработчику исключения, мы можем быть уверены, что ресурс освободится без нашего участия. Даже если ресурсов несколько, мы уверены, что все они будут корректно освобождены. В противном случае, если захватывать и освобождать ресурс вручную, то при возникновении исключения нужно учитывать, какие ресурсы уже захвачены, а какие – нет, и освобождать только использованные ресурсы, что не слишком просто.](http://virtser.net/blog/post/source-code-highlighter.aspx)
* [Идиома очень удобна, когда нужно отслеживать важные ресурсы, а при этом сопровождение кода оставляет желать лучшего.](http://virtser.net/blog/post/source-code-highlighter.aspx)
* [Часто при использовании нескольких ресурсов освобождать их следует в обратном порядке. При использовании идиомы RAII вследствие того, что объекты с захваченными ресурсами располагаются на стеке, их уничтожение происходит в обратном порядке, что как правило и является желательным.](http://virtser.net/blog/post/source-code-highlighter.aspx)
* [Поддержка принципа DRY (Don’t Repeat Yourself). Код инициализации и освобождения ресурса содержится только в одном месте. Нет необходимости копировать и вставлять код инициализации в каждое место в программе, где это необходимо. Достаточно просто создать объект.](http://virtser.net/blog/post/source-code-highlighter.aspx)
* [При необходимости использовать дополнительные параметры для обращения к ресурсу (например, логин и пароль к БД) эти параметры могут быть переданы в качестве аргументов конструктора.](http://virtser.net/blog/post/source-code-highlighter.aspx)
* [Накладные расходы при простейшей реализации обращения к ресурсу минимальны. В С++, как правило, при оптимизации компилятор реализует невиртуальные конструкторы и деструкторы в виде inline-функций.](http://virtser.net/blog/post/source-code-highlighter.aspx)
* [Данная идиома применима только в языках с предсказуемым временем жизни объекта. Сюда относится, например, С++, а также языки с сборщиком мусора, где время жизни объекта определяется количеством ссылок на него, такие как Objective C.](http://virtser.net/blog/post/source-code-highlighter.aspx)
* [Эта идиома неприменима в таких языках как Java или С#, где невозможно предсказать, когда объект будет удален.](http://virtser.net/blog/post/source-code-highlighter.aspx)

Источники:

1. <https://habr.com/ru/sandbox/21603/>
2. <https://www.cppstories.com/2016/04/custom-deleters-for-c-smart-pointers/>
3. <https://eax.me/cpp-smart-pointers/>
4. <https://ravesli.com/urok-189-umnye-ukazateli-i-semantika-peremeshheniya/>
5. <https://proglib.io/p/umnye-ukazateli-v-c-2021-08-26>
6. <https://proglib.io/p/umnye-ukazateli-v-c-2021-08-26>