Управление памятью в DLL C++

Статья

https://stackoverflow.com/questions/13625388/is-it-bad-practice-to-allocate-memory-in-a-dll-and-give-a-pointer-to-it-to-a-cli

Со stackoverflow

У C++ нету бинарного стандарта. Это значит, что один и тот же класс при компиляции разными компиляторами (да что там, даже при компиляции одним и тем же компилятором с разными флагами) может иметь разный бинарный layout. Может быть разное представление таблицы виртуальных методов, разная декорация имён методов, да просто разный физический размер. Поэтому передача между компонентами, скомпилированными разными компиляторами или разными версиями одного и того же компилятора, чего-то сложнее, чем простые структуры, закономерно может привести к проблемам и топтанию по чужой памяти. И в сложных случаях имеет смысл указывать `#pragma pack`.

Это объясняет, почему в экспортируемых из DLL заголовочных файлах часто встречается `extern "C"`.

Если DLL гарантированно скомпилирована той же версией компилятора с теми же ключами, в этом случае передавать объекты можно. Но это сразу же вносит проблемы с версионированием: обновление компонент возвращает проблему.

Это объясняет, почему не получится передавать объекты по смарт-указателю, если в проекте не все DLL скомпилированы той же версией компилятора. (А также почему системные DLL не используют смарт-указатели, исключения и другие фичи C++.)

Далее, о проблеме принадлежности памяти. Эта часть Microsoft-специфическая. Компиляция DLL имеет режимы со статической и динамической компоновкой рантайм-библиотеки (библиотеки языка c & c++ от Microsoft).

В случае статической компоновки рантайма у вас есть своя реализация `new` и `delete` в каждой DLL, они друг о друге не знают ничего, и обладают непересекающимися heap'ами. Это значит, что вы должны обеспечить, чтобы объект деаллоцировался тем же рантаймом, который его аллоцировал.

В случае динамической компоновки рантайма, у вас есть один рантайм на приложение — но только в случае, когда ваше приложение скомпилировано одной и той же версией студии! В этом хорошем случае вы можете аллоцировать объект в одной DLL, и деаллоцировать в другой. В плохом случае, когда у вас разные DLL скомпилированы разными версиями студии, они ссылаются на разные версии рантайм-библиотеки, и в этом случае у вас в бегущей программе будет по одному рантайму на каждую версию студии, с которой компилировались её DLL. С очевидными последствиями.