Subject: Re: Doubt with Buffer<> of a trivially destructible type
Posted by koldo on Sun, 15 Dec 2024 11:45:27 GMT
View Forum Message <> Reply to Message

The solution for U++ could be any of these, applied to class Buffer<> functions Malloc() and Free():

- For C++ 17, just insert constexpr in the std::is_trivially_destructible check, to force it to work in compile time:
if constexpr (std::is_trivially_destructible<T>::value)

- In C++ it is a little more complicated as it requires a little of SFINAE. This way the std::is_trivially_destructible check works in compile time:

```
template <typename U = T>
typename std::enable_if<!std::is_trivially_destructible<U>::value>::type
Malloc(size_t size) {
 void* p = MemoryAlloc(size * sizeof(U) + 16);
 *(size_t*)p = size;
 ptr = (U*)((byte*)p + 16);
}
template <typename U = T>
typename std::enable_if<std::is_trivially_destructible<U>::value>::type
Malloc(size_t size) {
 ptr = (U*)MemoryAlloc(size * sizeof(U));
}

template <typename U = T>
typename std::enable_if<!std::is_trivially_destructible<U>::value>::type
Free() {
 if (ptr) {
  void* p = (byte*)ptr - 16;
  size_t size = *(size_t*)p;
  Destroy(ptr, ptr + size);
  MemoryFree(p);
 }
}
template <typename U = T>
typename std::enable_if<std::is_trivially_destructible<U>::value>::type
Free() {
 if (ptr) {
  MemoryFree(ptr);
 }
}
```