Subject: Re: How to mark std::array<T, N> moveable if only T is moveable
Posted by Lance on Thu, 25 Jul 2024 22:00:16 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 25 July 2024 04:07
// For instance mark std::array<T, N> moveable if only T is moveable
template <class T, size_t N>
struct NtlMoveableClass<std::array<T, N>>
: std::integral_constant<bool, NtlMoveableClass<T>::value> {};

BTW, easiest and really nice way how to add optional trait:

template <> inline constexpr bool is_moveable<std::string> = true;

(do not want this, but could be useful to know)

Exactly, like it or not, a programmer can mark a class as moveable.

Another not very convincing example.

```
#include <iostream>
#include <type_traits>
#include <array>


template <class T> struct Moveable {};

template <class T>
inline constexpr bool is_moveable_v = std::is_trivially_copyable<T>::value ||
                                      std::is_base_of<Moveable<T>, T>::value;

class C: public Moveable<C>{
 C(const C& c){}
 C(C&& c){}
};

// uncomment to communicate the moveability of array<T,n> to the compiler
//template <class T, int n>
//inline constexpr bool is_moveable_v<std::array<T,n>> = is_moveable_v<T>;

int main()
{
 std::cout<<is_moveable_v<std::array<C,5>><<std::endl;
```

```
}
```