

Fixing constexpr member functions without const

ISO/IEC JTC1 SC22 WG21 N3669 - 2013-04-19

Nicolai Josuttis; nico@josuttis.de

Motivation:

Part of D3652 "Relaxing constraints on constexpr functions"

<http://wiki.edg.com/wiki/pub/Wg21bristol/CoreWorkingGroup/d3652.html>

is a breaking change in [dcl.constexpr] (7.1.5)/8, by **striking** the following sentence:

~~A constexpr specifier for a nonstatic member function that is not a constructor declares that member function to be const (9.3.1).~~

That is: **constexpr member functions are no longer implicitly const**

Thus, all member functions (except constructors) that were declared in C++11 as being constexpr, but not const, are no longer callable for constant objects. This might/will break a significantly amount of code.

To avoid this break, I suggest the following corresponding changes in the library, introducing const for each constexpr member function, where it doesn't occur yet.

Proposed changes:

In 20.5 Class template bitset [template.bitset]

class bitset

change:

constexpr size_t size() noexcept;

into:

constexpr size_t size() **const** noexcept;

In 20.5.2 bitset members [bitset.members]

change before paragraph 38:

constexpr size_t size() noexcept;

into

constexpr size_t size() **const** noexcept;

and change before paragraph 49:

constexpr bool operator[](size_t pos);

into

constexpr bool operator[](size_t pos) **const**;

In 20.9.3 Helper classes [meta.help]

in struct integral_constant change:

constexpr operator value_type() { return value; }

into:

constexpr operator value_type() **const** { return value; }

In 23.3.2.1 Class template array overview [array.overview]

in class array<> change:

```
constexpr size_type size() noexcept;  
constexpr size_type max_size() noexcept;  
constexpr bool empty() noexcept;
```

into:

```
constexpr size_type size() const noexcept;  
constexpr size_type max_size() const noexcept;  
constexpr bool empty() const noexcept;
```

In 23.3.2.4 array::size [array.size]

class array<> change:

```
template <class T, size_t N> constexpr size_type array<T,N>::size() noexcept;
```

into:

```
template <class T, size_t N> constexpr size_type array<T,N>::size() const noexcept;
```

In 26.4.3 complex specializations [complex.special]

in complex<float> change:

```
constexpr float real();  
void real(float);  
constexpr float imag();  
void imag(float);
```

into:

```
constexpr float real() const;  
void real(float);  
constexpr float imag() const;  
void imag(float);
```

and in complex<double> change:

```
constexpr double real();  
void real(double);  
constexpr double imag();  
void imag(double);
```

into:

```
constexpr double real() const;  
void real(double);  
constexpr double imag() const;  
void imag(double);
```

and in complex(long double> change:

```
constexpr long double real();  
void real(long double);  
constexpr long double imag();  
void imag(long double);
```

into:

```
constexpr long double real() const;  
void real(long double);
```

```
constexpr long double imag() const;  
void imag(long double);
```