

Document number: P2417R0  
Project: Programming Language C++  
Audience: LWG, LEWG

Daniil Goncharov [neargye@gmail.com](mailto:neargye@gmail.com)

Date: 2021-07-24

# A more constexpr bitset

---

## I. Changelog

Revision 0:

- Revised [P1251] after [discussed in Cologne](#), because the original author is no longer active and non-responsive.

## II. Motivation

`constexpr bitset` will allow to naturally use them as flags-mask in `constexpr/constexpr` functions. It's add, without limitations, new high-level and more user-friendly class for bit mask in embedded developing.

As of N4762 only the default constructor, the constructor accepting an `unsigned long long` and `operator[]` of `bitset` are marked as `constexpr`. With the adoption of [P0980](#), there is no reason the rest of the class cannot be made `constexpr`. The lack of `constexpr` for most member functions was probably due to the nontrivial destructor of `bitset::reference`. Now it is possible to mark the destructor and therefore the rest of `bitset` as `constexpr` instead of requiring trivial destructability of `bitset::reference` and risking potential ABI breaks in certain implementations.

## III. Proposed Changes

Mark every member function except iostream operators. Make all of `bitset::reference` `constexpr`.

## IV. Impact on the Standard

This proposal is a pure library addition.

## V. Proposed wording

### A. Modifications to "20.9 Bitsets" [bitset]

All the additions to the Standard are marked with **green**.

#### **Change 20.9.1 of N4892 to the following:**

```
#include <string>
#include <iosfwd> // for istream (29.7.1), ostream (29.7.2), see 29.3.1

namespace std {
template<size_t N> class bitset;

// 20.9.4, bitset operators
template<size_t N>
    constexpr bitset<N> operator&(const bitset<N>&, const bitset<N>&)
noexcept;

template<size_t N>
    constexpr bitset<N> operator|(const bitset<N>&, const bitset<N>&)
noexcept;

template<size_t N>
    constexpr bitset<N> operator^(const bitset<N>&, const bitset<N>&)
noexcept;

template<class charT, class traits, size_t N>
    basic_istream<charT, traits>&
        operator>>(basic_istream<charT, traits>& is, bitset<N>& x);

template<class charT, class traits, size_t N>
    basic_ostream<charT, traits>&
        operator<<(basic_ostream<charT, traits>& os, const bitset<N>& x);
}
```

## Change 20.9.2.1 of N4892 to the following:

```
namespace std {
    template<size_t N> class bitset {
    public:
        // bit reference
        class reference {
            friend class bitset;
            constexpr reference() noexcept;

        public:
            constexpr reference(const reference&) = default;
            constexpr ~reference();
            constexpr reference& operator=(bool x) noexcept;           // for
b[i] = x;
            constexpr reference& operator=(const reference&) noexcept; // for
b[i] = b[j];
            constexpr bool operator~() const noexcept;               // flips
the bit
            constexpr operator bool() const noexcept;                // for x
= b[i];
            constexpr reference& flip() noexcept;                     // for
b[i].flip();
        };

        // 19.9.2.1, constructors
        constexpr bitset() noexcept;
        constexpr bitset(unsigned long long val) noexcept;
        template<class charT, class traits, class Allocator>
            constexpr explicit bitset(
                const basic_string<charT, traits, Allocator>& str,
                typename basic_string<charT, traits, Allocator>::size_type pos = 0,
                typename basic_string<charT, traits, Allocator>::size_type n
                    = basic_string<charT, traits, Allocator>::npos,
                charT zero = charT('0'),
                charT one = charT('1'));
        template<class charT>
            constexpr explicit bitset(
                const charT* str,
                typename basic_string<charT>::size_type n =
basic_string<charT>::npos,
                charT zero = charT('0'),
                charT one = charT('1'));

        // 19.9.2.2, bitset operations
        constexpr bitset<N>& operator&=(const bitset<N>& rhs) noexcept;
        constexpr bitset<N>& operator|=(const bitset<N>& rhs) noexcept;
        constexpr bitset<N>& operator^=(const bitset<N>& rhs) noexcept;
        constexpr bitset<N>& operator<<=(size_t pos) noexcept;
        constexpr bitset<N>& operator>>=(size_t pos) noexcept;
        constexpr bitset<N>& set() noexcept;
    };
};
```

```

constexpr bitset<N>& set(size_t pos, bool val = true);
constexpr bitset<N>& reset() noexcept;
constexpr bitset<N>& reset(size_t pos);
constexpr bitset<N> operator~() const noexcept;
constexpr bitset<N>& flip() noexcept;
constexpr bitset<N>& flip(size_t pos);

// element access
constexpr bool operator[](size_t pos) const;           // for b[i];
constexpr reference operator[](size_t pos);           // for b[i];

constexpr unsigned long to_ulong() const;
constexpr unsigned long long to_ullong() const;
template<class charT = char,
        class traits = char_traits<charT>,
        class Allocator = allocator<charT>>
constexpr basic_string<charT, traits, Allocator>
    to_string(charT zero = charT('0'), charT one = charT('1')) const;

constexpr size_t count() const noexcept;
constexpr constexpr size_t size() const noexcept;
constexpr bool operator==(const bitset<N>& rhs) const noexcept;
constexpr bool operator!=(const bitset<N>& rhs) const noexcept;
constexpr bool test(size_t pos) const;
constexpr bool all() const noexcept;
constexpr bool any() const noexcept;
constexpr bool none() const noexcept;
constexpr bitset<N> operator<<(size_t pos) const noexcept;
constexpr bitset<N> operator>>(size_t pos) const noexcept;
};

// 19.9.3, hash support
template<class T> struct hash;
template<size_t N> struct hash<bitset<N>>;
}

```

## B. Modify to "17.3.2 Header <version> synopsis" [version.syn]

```
#define __cpp_lib_constexpr_bitset DATE OF ADOPTION // also in <bitset>
```

## VI. Acknowledgements

Thanks to Morris Hafner for the work done on the original version of the paper [P1251].

## VII. References

- [n4892] Working Draft, Standard for Programming Language C++. Available online at <https://github.com/cplusplus/draft/releases/download/n4892/n4892.pdf>
- [P1251] Morris Hafner: A more constexpr bitset <https://wg21.link/p1251>
- [P0784] L. Dionne, R. Smith, N. Ranns, D.Vandevoorde: More constexpr containers <https://wg21.link/p0784>

- [P0980] L. Dionne: Making std::string constexpr <https://wg21.link/p0980>