On the names of low-level bit manipulation functions

Abstract

We provide wording to rename the bit functions following National Body comments PL326, US327, GB332, US328, GB331 and following LEWG votes in Belfast 2019.

Contents

Introduction 1
Wording 2
Acknowledgements 4

Introduction

In Belfast, and following several national body comments and SG6 suggestions, LEWG voted to rename some of the bit manipulation functions in order to avoid references to powers of two since manipulating bits should work in the future on bytes and machine words for which numeric operations are not defined.
0.1 Bit manipulation

0.1.1 General

The header bit provides components to access, manipulate and process both individual bits and bit sequences. Binary representation is assumed as in ISO/IEC 80000-13 item 13-9.

0.1.2 Header <bit> synopsis

namespace std {

    template<class To, class From>
        constexpr To bit_cast(const From& from) noexcept;

    template<class T>
        constexpr bool has_single_bit(T x) noexcept;

    template<class T>
        constexpr T bit_ceil(T x);

    template<class T>
        constexpr T bit_floor(T x) noexcept;

    template<class T>
        constexpr T bit_width(T x) noexcept;

    template<class T>
        [[nodiscard]] constexpr T rotl(T x, int s) noexcept;

    template<class T>
        [[nodiscard]] constexpr T rotr(T x, int s) noexcept;

    template<class T>
        constexpr int countl_zero(T x) noexcept;

    template<class T>
        constexpr int countl_one(T x) noexcept;

    template<class T>
        constexpr int countr_zero(T x) noexcept;

    template<class T>
        constexpr int countr_one(T x) noexcept;

    template<class T>
        constexpr int popcount(T x) noexcept;

    enum class endian {
        little = see below,
        big   = see below,
        native = see below
    };
}

0.1.3 Function template bit_cast

Nothing to modify.

0.1.4 Single bit manipulation

has_single_bit

template<class T>
constexpr bool has_single_bit(T x) noexcept;

T is an unsigned integer type basic.fundamental.

Returns: true if \( x \) is an integral power of two (has only one bit set); false otherwise.

bit_ceil

\[ \text{template<class T>} \]
\[ \text{constexpr T bit_ceil(T x);} \]

Let \( N \) be the smallest power of 2 greater than or equal to \( x \).

\( N \) is representable as a value of type \( T \).

Returns: \( N \).

Throws: Nothing.

Remarks: A function call expression that violates the precondition in the element is not a core constant expression expr.const.

bit_floor

\[ \text{template<class T>} \]
\[ \text{constexpr T bit_floor(T x) noexcept;} \]

\( T \) is an unsigned integer type basic.fundamental.

Returns: If \( x == 0 \), 0; otherwise the maximal value \( y \) such that has_single_bit(\( y \)) is true and \( y \leq x \).

If \( x == 0 \), 0; otherwise \( x \) where all the bits except the most significant one have been cleared.

bit_width

\[ \text{template<class T>} \]
\[ \text{constexpr T bit_width(T x) noexcept;} \]

\( T \) is an unsigned integer type basic.fundamental.

Returns: If \( x == 0 \), 0; otherwise one plus the base-2 logarithm of \( x \), with any fractional part discarded.

0.1.5 Rotating [bit.rotate]

Nothing to modify.

0.1.6 Counting [bit.count]

Nothing to modify.

0.1.7 Endian [bit.endian]

Nothing to modify.
Acknowledgements
This work has been made possible thanks to the National Science Foundation through the awards CCF-1647432 and SI2-SSE-1642411, as well as French institutions École Normale Supérieure, INRIA, and PSL.