Stdc .h

C23 (C standard revision)

type-generic macro. Add stdc_count_ones() and stdc_count_zeros*() to count number of 1 or 0 bits in value. Add stdc_leading_ones*() and stdc_leading_zeros*()*

C23, formally ISO/IEC 9899:2024, is the current open standard for the C programming language, which supersedes C17 (standard ISO/IEC 9899:2018). It was started in 2016 informally as C2x, and was published on October 31, 2024. The freely available draft most similar to the one published is document N3220 (see Available texts, below). The first WG14 meeting for the C2x draft was held in October 2019, virtual remote meetings were held in 2020 due to the COVID-19 pandemic, then various teleconference meetings continued to occur through 2024.

In C23, the value of __STDC_VERSION__ changes from 201710L to 202311L. The common names "C17" and "C23" reflect these values, which are frozen prior to final adoption, rather than the years in the ISO standards identifiers (9899:2018 and 9899:2024).

C99

#ifndef __STDC_IEC_559__ puts("Warning: __STDC_IEC_559__ not defined. IEEE 754 floating point not fully supported."); // [9] #endif #pragma STDC FENV_ACCESS

C99 (C9X during its development, formally ISO/IEC 9899:1999) is a past version of the C programming language open standard. It extends the previous version (C90) with new features for the language and the standard library, and helps implementations make better use of available computer hardware, such as IEEE 754-1985 floating-point arithmetic, and compiler technology. The C11 version of the C programming language standard, published in 2011, updates C99.

ANSI C

2:1996 in 1996 #if defined(__STDC_VERSION__) && __STDC_VERSION__ >= 199409L / C95 compatible source code. */ #elif defined(__STDC__) /* C89 compatible source*

ANSI C, ISO C, and Standard C are successive standards for the C programming language published by the American National Standards Institute (ANSI) and ISO/IEC JTC 1/SC 22/WG 14 of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). Historically, the names referred specifically to the original and best-supported version of the standard (known as C89 or C90). Software developers writing in C are encouraged to conform to the standards, as doing so helps portability between compilers.

Stoer–Wagner algorithm

Stoer–Wagner min cut algorithm. // // Running time: // O(|V|^3) #include <bits/stdc++.h> using namespace std; pair<int, vector<int>> globalMinCut(vector<vector<int>>

In graph theory, the Stoer–Wagner algorithm is a recursive algorithm to solve the minimum cut problem in undirected weighted graphs with non-negative weights. It was proposed by Mechthild Stoer and Frank Wagner in 1995. The essential idea of this algorithm is to shrink the graph by merging the most intensive vertices, until the graph only contains two combined vertex sets. At each phase, the algorithm finds the minimum

s

{\displaystyle s}

-

t

{\displaystyle t}

cut for two vertices

s

{\displaystyle s}

and

t

{\displaystyle t}

chosen at its will. Then the algorithm shrinks the edge between

s

{\displaystyle s}

and

t

{\displaystyle t}

to search for non

s

{\displaystyle s}

-

t

{\displaystyle t}

cuts. The minimum cut found in all phases will be the minimum weighted cut of the graph.

A cut is a partition of the vertices of a graph into two non-empty, disjoint subsets. A minimum cut is a cut for which the size or weight of the cut is not larger than the size of any other cut. For an unweighted graph, the minimum cut would simply be the cut with the least edges. For a weighted graph, the sum of all edges' weight on the cut determines whether it is a minimum cut. In practice, the minimum cut problem is always discussed with the maximum flow problem, to explore the maximum capacity of a network, since the minimum cut is a bottleneck in a graph or network.

C11 (C standard revision)

requiring resolution by participating national bodies. A standard macro __STDC_VERSION__ is defined with value 201112L to indicate that C11 support is available

C11 (previously C1X, formally ISO/IEC 9899:2011) is a past standard for the C programming language. It replaced C99 (standard ISO/IEC 9899:1999) and has been superseded by C17 (standard ISO/IEC 9899:2018). C11 mainly standardizes features already supported by common contemporary compilers, and includes a detailed memory model to better support multiple threads of execution. Due to delayed availability of conforming C99 implementations, C11 makes certain features optional, to make it easier to comply with the core language standard.

The final draft, N1570, was published in April 2011. The new standard passed its final draft review on October 10, 2011 and was officially ratified by ISO and published as ISO/IEC 9899:2011 on December 8, 2011, with no comments requiring resolution by participating national bodies.

A standard macro __STDC_VERSION__ is defined with value 201112L to indicate that C11 support is available.

C standard library

C++?11 <stdbool.h>/cstdbool> headers are fully equivalent. (In particular, C23 doesn't require any __STDC_VERSION_BOOL_H__ macro for <stdbool.h>.) Access to

The C standard library, sometimes referred to as libc, is the standard library for the C programming language, as specified in the ISO C standard. Starting from the original ANSI C standard, it was developed at the same time as the C POSIX library, which is a superset of it. Since ANSI C was adopted by the International Organization for Standardization, the C standard library is also called the ISO C library.

The C standard library provides macros, type definitions and functions for tasks such as string manipulation, mathematical computation, input/output processing, memory management, and input/output.

C (programming language)

compilable by either standard-conforming or K&R C-based compilers, the __STDC__ macro can be used to split the code into Standard and K&R sections to prevent

C is a general-purpose programming language. It was created in the 1970s by Dennis Ritchie and remains widely used and influential. By design, C gives the programmer relatively direct access to the features of the typical CPU architecture, customized for the target instruction set. It has been and continues to be used to implement operating systems (especially kernels), device drivers, and protocol stacks, but its use in application software has been decreasing. C is used on computers that range from the largest supercomputers to the smallest microcontrollers and embedded systems.

A successor to the programming language B, C was originally developed at Bell Labs by Ritchie between 1972 and 1973 to construct utilities running on Unix. It was applied to re-implementing the kernel of the Unix operating system. During the 1980s, C gradually gained popularity. It has become one of the most widely used programming languages, with C compilers available for practically all modern computer architectures and operating systems. The book The C Programming Language, co-authored by the original language designer, served for many years as the de facto standard for the language. C has been standardized since 1989 by the American National Standards Institute (ANSI) and, subsequently, jointly by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC).

C is an imperative procedural language, supporting structured programming, lexical variable scope, and recursion, with a static type system. It was designed to be compiled to provide low-level access to memory

and language constructs that map efficiently to machine instructions, all with minimal runtime support. Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A standards-compliant C program written with portability in mind can be compiled for a wide variety of computer platforms and operating systems with few changes to its source code.

Although neither C nor its standard library provide some popular features found in other languages, it is flexible enough to support them. For example, object orientation and garbage collection are provided by external libraries GLib Object System and Boehm garbage collector, respectively.

Since 2000, C has consistently ranked among the top four languages in the TIOBE index, a measure of the popularity of programming languages.

Multiply–accumulate operation

which can be explicitly enabled or disabled with standard pragmas (#pragma STDC FP_CONTRACT). The GCC and Clang C compilers do such transformations by default

In computing, especially digital signal processing, the multiply–accumulate (MAC) or multiply–add (MAD) operation is a common step that computes the product of two numbers and adds that product to an accumulator. The hardware unit that performs the operation is known as a multiplier–accumulator (MAC unit); the operation itself is also often called a MAC or a MAD operation. The MAC operation modifies an accumulator a:

a

?

a

+

(

b

×

c

)

{\displaystyle a\gets a+(b\times c)}

When done with floating-point numbers, it might be performed with two roundings (typical in many DSPs), or with a single rounding. When performed with a single rounding, it is called a fused multiply–add (FMA) or fused multiply–accumulate (FMAC).

Modern computers may contain a dedicated MAC, consisting of a multiplier implemented in combinational logic followed by an adder and an accumulator register that stores the result. The output of the register is fed back to one input of the adder, so that on each clock cycle, the output of the multiplier is added to the register. Combinational multipliers require a large amount of logic, but can compute a product much more quickly than the method of shifting and adding typical of earlier computers. Percy Ludgate was the first to conceive a MAC in his Analytical Machine of 1909, and the first to exploit a MAC for division (using multiplication seeded by reciprocal, via the convergent series (1+x)?1). The first modern processors to be equipped with MAC units were digital signal processors, but the technique is now also common in general-

purpose processors.

C preprocessor

Standard specified that __STDC__ expand to "1" if the implementation conforms to the ISO standard and "0" otherwise and that __STDC_VERSION__ expand to a

The C preprocessor (CPP) is a text file processor that is used with C, C++ and other programming tools. The preprocessor provides for file inclusion (often header files), macro expansion, conditional compilation, and line control. Although named in association with C and used with C, the preprocessor capabilities are not inherently tied to the C language. It can and is used to process other kinds of files.

C, C++, and Objective-C compilers provide a preprocessor capability, as it is required by the definition of each language. Some compilers provide extensions and deviations from the target language standard. Some provide options to control standards compliance. For instance, the GNU C preprocessor can be made more standards compliant by supplying certain command-line flags.

The C# programming language also allows for directives, even though they cannot be used for creating macros, and is generally more intended for features such as conditional compilation. C# seldom requires the use of the directives, for example code inclusion does not require a preprocessor at all (as C# relies on a package/namespace system like Java, no code needs to be "included").

The Haskell programming language also allows the usage of the C preprocessor, which is invoked by writing {-# LANGUAGE CPP #-} at the top of the file. The accepted preprocessor directives align with those in standard C/C++.

Features of the preprocessor are encoded in source code as directives that start with #.

Although C++ source files are often named with a .cpp extension, that is an abbreviation for "C plus plus"; not C preprocessor.

Subnormal number

It is known to work on Mac OS X since at least 2006. #include <fenv.h> #pragma STDC FENV_ACCESS ON // Sets DAZ and FTZ, clobbering other CSR settings.

In computer science, subnormal numbers are the subset of denormalized numbers (sometimes called denormals) that fill the underflow gap around zero in floating-point arithmetic. Any non-zero number with magnitude smaller than the smallest positive normal number is subnormal, while denormal can also refer to numbers outside that range.