


















CC. Undefined Behavior

Created by Martin Sebor, last modified by David Svoboda on Jun 21, 2018
















According to the C Standard, Annex J, J.2 [ISO/IEC 9899:2011], the behavior of a program is [undefined](#) in the circumstances outlined in the following table. The "Guideline" column in the table identifies the coding practices that address the specific case of undefined behavior (UB). The descriptions of undefined behaviors in the "Description" column are direct quotes from the standard. The parenthesized numbers refer to the subclause of the C Standard (C11) that identifies the undefined behavior.

UB	Class	Description	Guideline
1	✖	A "shall" or "shall not" requirement that appears outside of a constraint is violated (clause 4).	MSC15-C
2	⚠	A nonempty source file does not end in a new-line character which is not immediately preceded by a backslash character or ends in a partial preprocessing token or comment (5.1.1.2).	
3	⚠	Token concatenation produces a character sequence matching the syntax of a universal character name (5.1.1.2).	PRE30-C
4	⚠	A program in a hosted environment does not define a function named <code>main</code> using one of the specified forms (5.1.2.2.1).	
5		The execution of a program contains a data race (5.1.2.4).	
6	⚠	A character not in the basic source character set is encountered in a source file, except in an identifier, a character constant, a string literal, a header name, a comment, or a preprocessing token that is never converted to a token (5.2.1).	
7	⚠	An identifier, comment, string literal, character constant, or header name contains an invalid multibyte character or does not begin and end in the initial shift state (5.2.1.2).	
8	⚠	The same identifier has both internal and external linkage in the same translation unit (6.2.2).	DCL36-C
9	✖	An object is referred to outside of its lifetime (6.2.4).	DCL21-C, DCL30-C
10	✖	The value of a pointer to an object whose lifetime has ended is used (6.2.4).	DCL30-C, EXP33-C
11	⚠	The value of an object with automatic storage duration is used while it is indeterminate (6.2.4, 6.7.9, 6.8).	EXP33-C, MSC22-C
12	⚠	A trap representation is read by an lvalue expression that does not have character type (6.2.6.1).	EXP33-C

UB	Class	Description	Guideline
14		The operands to certain operators are such that they could produce a negative zero result, but the implementation does not support negative zeros (6.2.6.2).	
15		Two declarations of the same object or function specify types that are not compatible (6.2.7).	DCL23-C , DCL40-C
16		A program requires the formation of a composite type from a variable length array type whose size is specified by an expression that is not evaluated (6.2.7).	
17		Conversion to or from an integer type produces a value outside the range that can be represented (6.3.1.4).	FLP34-C
18		Demotion of one real floating type to another produces a value outside the range that can be represented (6.3.1.5).	FLP34-C
19		An lvalue does not designate an object when evaluated (6.3.2.1).	
20		A non-array lvalue with an incomplete type is used in a context that requires the value of the designated object (6.3.2.1).	
21		An lvalue designation an object of automatic storage duration that could have been declared with the <code>register</code> storage class is used in a context that requires the value of the designated object, but the object is uninitialized (6.3.2.1).	
22		An lvalue having array type is converted to a pointer to the initial element of the array, and the array object has <code>register</code> storage class (6.3.2.1).	
23		An attempt is made to use the value of a void expression, or an implicit or explicit conversion (except to <code>void</code>) is applied to a void expression (6.3.2.2).	
24		Conversion of a pointer to an integer type produces a value outside the range that can be represented (6.3.2.3).	INT36-C
25		Conversion between two pointer types produces a result that is incorrectly aligned (6.3.2.3).	EXP36-C
26		A pointer is used to call a function whose type is not compatible with the pointed-to type (6.3.2.3).	EXP37-C
27		An unmatched ' or " character is encountered on a logical source line during tokenization (6.4).	
28		A reserved keyword token is used in translation phase 7 or 8 for some purpose other than as a keyword (6.4.1).	
29		A universal character name in an identifier does not designate a character whose encoding falls into one of the specified ranges (6.4.2.1).	
30		The initial character of an identifier is a universal character name designating a digit (6.4.2.1).	
31		Two identifiers differ only in nonsignificant characters (6.4.2.1).	DCL23-C , DCL31-C
32		The identifier <code>func{ }</code> is explicitly declared (6.4.2.2).	



















UB	Class	Description	Guideline
		sequence between the " delimiters, in a header name preprocessing token (6.4.7).	
35	⚠	A side effect on a scalar object is unsequenced relative to either a different side effect on the same scalar object or a value computation using the value of the same scalar object (6.5).	EXP30-C
36	⚠	An exceptional condition occurs during the evaluation of an expression (6.5).	INT32-C
37	⚠	An object has its stored value accessed other than by an lvalue of an allowable type (6.5).	DCL40-C, EXP39-C
38	⚠	For a call to a function without a function prototype in scope, the number of arguments does not equal the number of parameters (6.5.2.2).	EXP37-C
39	⚠	For call to a function without a function prototype in scope where the function is defined with a function prototype, either the prototype ends with an ellipsis or the types of the arguments after promotion are not compatible with the types of the parameters (6.5.2.2).	EXP37-C
40	⚠	For a call to a function without a function prototype in scope where the function is not defined with a function prototype, the types of the arguments after promotion are not compatible with those of the parameters after promotion (with certain exceptions) (6.5.2.2).	EXP37-C
41	⚠	A function is defined with a type that is not compatible with the type (of the expression) pointed to by the expression that denotes the called function (6.5.2.2).	DCL40-C, EXP37-C
42		A member of an atomic structure or union is accessed (6.5.2.3).	
43	✖	The operand of the unary * operator has an invalid value (6.5.3.2).	
44	⚠	A pointer is converted to other than an integer or pointer type (6.5.4).	
45	⚠	The value of the second operand of the / or % operator is zero (6.5.5).	INT33-C
46	⚠	Addition or subtraction of a pointer into, or just beyond, an array object and an integer type produces a result that does not point into, or just beyond, the same array object (6.5.6).	ARR30-C
47	✖	Addition or subtraction of a pointer into, or just beyond, an array object and an integer type produces a result that points just beyond the array object and is used as the operand of a unary * operator that is evaluated (6.5.6).	ARR30-C
48	⚠	Pointers that do not point into, or just beyond, the same array object are subtracted (6.5.6).	ARR36-C
49	⚠	An array subscript is out of range, even if an object is apparently accessible with the given subscript (as in the lvalue expression <code>a[1][7]</code> given the declaration <code>int a[4][5]</code>) (6.5.6).	ARR30-C
50	⚠	The result of subtracting two pointers is not representable in an object of type <code>ptrdiff_t</code> (6.5.6).	


















UB	Class	Description	Guideline
		representable in the promoted type (6.5.7).	
53	⚠	Pointers that do not point to the same aggregate or union (nor just beyond the same array object) are compared using relational operators (6.5.8).	ARR36-C
54	⚠	An object is assigned to an inexact overlapping object or to an exactly overlapping object with incompatible type (6.5.16.1).	
55	⚠	An expression that is required to be an integer constant expression does not have an integer type; has operands that are not integer constants, enumeration constants, character constants, <code>sizeof</code> expressions whose results are integer constants, or immediately-cast floating constants; or contains casts (outside operands to <code>sizeof</code> operators) other than conversions of arithmetic types to integer types (6.6).	
56	⚠	A constant expression in an initializer is not, or does not evaluate to, one of the following: an arithmetic constant expression, a null pointer constant, an address constant, or an address constant for an object type plus or minus an integer constant expression (6.6).	
57	⚠	An arithmetic constant expression does not have arithmetic type; has operands that are not integer constants, floating constants, enumeration constants, character constants, or <code>sizeof</code> expressions; or contains casts (outside operands to <code>sizeof</code> operators) other than conversions of arithmetic types to arithmetic types (6.6).	
58	⚠	The value of an object is accessed by an array-subscript <code>[]</code> , member-access <code>.</code> or <code>-></code> , address <code>&</code> , or indirection <code>*</code> operator or a pointer cast in creating an address constant (6.6).	
59	⚠	An identifier for an object is declared with no linkage and the type of the object is incomplete after its declarator, or after its init-declarator if it has an initializer (6.7).	
60	⚠	A function is declared at block scope with an explicit storage-class specifier other than <code>extern</code> (6.7.1).	
61	⚠	A structure or union is defined as containing no named members (6.7.2.1).	
62	⚠	An attempt is made to access, or generate a pointer to just past, a flexible array member of a structure when the referenced object provides no elements for that array (6.7.2.1).	ARR30-C
63	⚠	When the complete type is needed, an incomplete structure or union type is not completed in the same scope by another declaration of the tag that defines the content (6.7.2.3).	
64	ℹ	An attempt is made to modify an object defined with a <code>const</code> -qualified type through use of an lvalue with non- <code>const</code> -qualified type (6.7.3).	EXP05-C, EXP40-C
65	⚠	An attempt is made to refer to an object defined with a <code>volatile</code> -qualified type through use of an lvalue with non- <code>volatile</code> -qualified type (6.7.3).	EXP32-C
66	⚠	The specification of a function type includes any type qualifiers (6.7.3).	
67	⚠	Two qualified types that are required to be compatible do not have the identically qualified version of a compatible type (6.7.3).	














UB	Class	Description	Guideline
69		A <code>restrict</code> -qualified pointer is assigned a value based on another restricted pointer whose associated block neither began execution before the block associated with this pointer, nor ended before the assignment (6.7.3.1).	
70		A function with external linkage is declared with an <code>inline</code> function specifier, but is not also defined in the same translation unit (6.7.4).	
71		A function declared with a <code>_Noreturn</code> function specifier returns to its caller (6.7.4).	
72		The definition of an object has an alignment specifier and another declaration of that object has a different alignment specifier (6.7.5).	
73		Declarations of an object in different translation units have different alignment specifiers (6.7.5).	
74		Two pointer types that are required to be compatible are not identically qualified, or are not pointers to compatible types (6.7.6.1).	
75		The size expression in an array declaration is not a constant expression and evaluates at program execution time to a nonpositive value (6.7.6.2).	ARR32-C
76		In a context requiring two array types to be compatible, they do not have compatible element types, or their size specifiers evaluate to unequal values (6.7.6.2).	
77		A declaration of an array parameter includes the keyword <code>static</code> within the <code>[</code> and <code>]</code> and the corresponding argument does not provide access to the first element of an array with at least the specified number of elements (6.7.6.3).	
78		A storage-class specifier or type qualifier modifies the keyword <code>void</code> as a function parameter type list (6.7.6.3).	
79		In a context requiring two function types to be compatible, they do not have compatible return types, or their parameters disagree in use of the ellipsis terminator or the number and type of parameters (after default argument promotion, when there is no parameter type list or when one type is specified by a function definition with an identifier list) (6.7.6.3).	
80		The value of an unnamed member of a structure or union is used (6.7.9).	
81		The initializer for a scalar is neither a single expression nor a single expression enclosed in braces (6.7.9).	
82		The initializer for a structure or union object that has automatic storage duration is neither an initializer list nor a single expression that has compatible structure or union type (6.7.9).	
83		The initializer for an aggregate or union, other than an array initialized by a string literal, is not a brace-enclosed list of initializers for its elements or members (6.7.9).	
84		An identifier with external linkage is used, but in the program there does not exist exactly one external definition for the identifier, or the identifier is not used and there exist multiple external definitions for the identifier (6.9).	
85		A function definition includes an identifier list, but the types of the parameters are not declared in a following declaration list (6.9.1).	
86		An adjusted parameter type in a function definition is not an object type (6.9.1).	

UB	Class	Description	Guideline
----	-------	-------------	-----------

[SEI CERT C Coding Standard](#) / [4 Back Matter](#)
















89		An identifier for an object with internal linkage and an incomplete type is declared with a tentative definition (6.9.2).	
90		The token defined is generated during the expansion of a <code>#if</code> or <code>#elif</code> preprocessing directive, or the use of the defined unary operator does not match one of the two specified forms prior to macro replacement (6.10.1).	
91		The <code>#include</code> preprocessing directive that results after expansion does not match one of the two header name forms (6.10.2).	
92		The character sequence in an <code>#include</code> preprocessing directive does not start with a letter (6.10.2).	
93		There are sequences of preprocessing tokens within the list of macro arguments that would otherwise act as preprocessing directives (6.10.3).	PRE32-C
94		The result of the preprocessing operator <code>#</code> is not a valid character string literal (6.10.3.2).	
95		The result of the preprocessing operator <code>##</code> is not a valid preprocessing token (6.10.3.3).	
96		The <code>#line</code> preprocessing directive that results after expansion does not match one of the two well-defined forms, or its digit sequence specifies zero or a number greater than 2147483647 (6.10.4).	
97		A non- <code>STDC</code> <code>#pragma</code> preprocessing directive that is documented as causing translation failure or some other form of undefined behavior is encountered (6.10.6).	
98		A <code>#pragma STDC</code> preprocessing directive does not match one of the well-defined forms (6.10.6).	
99		The name of a predefined macro, or the identifier defined, is the subject of a <code>#define</code> or <code>#undef</code> preprocessing directive (6.10.8).	
100		An attempt is made to copy an object to an overlapping object by use of a library function, other than as explicitly allowed (e.g., <code>memcpy</code>) (clause 7).	
101		A file with the same name as one of the standard headers, not provided as part of the implementation, is placed in any of the standard places that are searched for included source files (7.1.2).	
102		A header is included within an external declaration or definition (7.1.2).	
103		A function, object, type, or macro that is specified as being declared or defined by some standard header is used before any header that declares or defines it is included (7.1.2).	
104		A standard header is included while a macro is defined with the same name as a keyword (7.1.2).	
105		The program attempts to declare a library function itself, rather than via a standard header, but the declaration does not have external linkage (7.1.2).	
106		The program declares or defines a reserved identifier, other than as allowed by 7.1.4 (7.1.3).	DCL37-C
















UB	Class	Description	Guideline
109		The pointer passed to a library function array parameter does not have a value such that all address computations and object accesses are valid (7.1.4).	ARR30-C , ARR38-C
110		The macro definition of <code>assert</code> is suppressed in order to access an actual function (7.2).	MSC38-C
111		The argument to the <code>assert</code> macro does not have a scalar type (7.2).	
112		The <code>CX_LIMITED_RANGE</code> , <code>FENV_ACCESS</code> , or <code>FP_CONTRACT</code> pragma is used in any context other than outside all external declarations or preceding all explicit declarations and statements inside a compound statement (7.3.4, 7.6.1, 7.12.2).	
113		The value of an argument to a character handling function is neither equal to the value of <code>EOF</code> nor representable as an <code>unsigned char</code> (7.4).	STR37-C
114		A macro definition of <code>errno</code> is suppressed in order to access an actual object, or the program defines an identifier with the name <code>errno</code> (7.5).	DCL37-C , MSC38-C
115		Part of the program tests floating-point status flags, sets floating-point control modes, or runs under non-default mode settings, but was translated with the state for the <code>FENV_ACCESS</code> pragma "off" (7.6.1).	
116		The exception-mask argument for one of the functions that provide access to the floating-point status flags has a nonzero value not obtained by bitwise OR of the floating-point exception macros (7.6.2).	
117		The <code>fesetexceptflag</code> function is used to set floating-point status flags that were not specified in the call to the <code>fegetexceptflag</code> function that provided the value of the corresponding <code>fexcept_t</code> object (7.6.2.4).	
118		The argument to <code>fesetenv</code> or <code>feupdateenv</code> is neither an object set by a call to <code>fegetenv</code> or <code>feholdexcept</code> , nor is it an environment macro (7.6.4.3, 7.6.4.4).	
119		The value of the result of an integer arithmetic or conversion function cannot be represented (7.8.2.1, 7.8.2.2, 7.8.2.3, 7.8.2.4, 7.22.6.1, 7.22.6.2, 7.22.1).	ERR07-C
120		The program modifies the string pointed to by the value returned by the <code>setlocale</code> function (7.11.1.1).	ENV30-C
121		The program modifies the structure pointed to by the value returned by the <code>localeconv</code> function (7.11.2.1).	ENV30-C
122		A macro definition of <code>math_errhandling</code> is suppressed or the program defines an identifier with the name <code>math_errhandling</code> (7.12).	MSC38-C
123		An argument to a floating-point classification or comparison macro is not of real floating type (7.12.3, 7.12.14).	
124		A macro definition of <code>setjmp</code> is suppressed in order to access an actual function, or the program defines an external identifier with the name <code>setjmp</code> (7.13).	MSC38-C
125		An invocation of the <code>setjmp</code> macro occurs other than in an allowed context (7.13.2.1).	MSC22-C

















UB	Class	Description	Guideline
		containing the invocation of the corresponding <code>setjmp</code> macro, that was changed between the <code>setjmp</code> invocation and <code>longjmp</code> call (7.13.2.1).	
128		The program specifies an invalid pointer to a signal handler function (7.14.1.1).	
129		A signal handler returns when the signal corresponded to a computational exception (7.14.1.1).	SIG31-C
130		A signal handler called in response to <code>SIGFPE</code> , <code>SIGILL</code> , <code>SIGSEGV</code> , or any other implementation-defined value corresponding to a computational exception returns (7.14.1.1).	SIG35-C
131		A signal occurs as the result of calling the <code>abort</code> or <code>raise</code> function, and the signal handler calls the <code>raise</code> function (7.14.1.1).	SIG30-C , SIG31-C
132		A signal occurs other than as the result of calling the <code>abort</code> or <code>raise</code> function, and the signal handler refers to an object with static or thread storage duration that is not a lock-free atomic object other than by assigning a value to an object declared as volatile <code>sig_atomic_t</code> , or calls any function in the standard library other than the <code>abort</code> function, the <code>_Exit</code> function, the <code>quick_exit</code> function, or the <code>signal</code> function (for the same signal number) (7.14.1.1).	SIG31-C
133		The value of <code>errno</code> is referred to after a signal occurred other than as the result of calling the <code>abort</code> or <code>raise</code> function and the corresponding signal handler obtained a <code>SIG_ERR</code> return from a call to the <code>signal</code> function (7.14.1.1).	ERR32-C
134		A signal is generated by an asynchronous signal handler (7.14.1.1).	
135		The <code>signal</code> function is used in a multi-threaded program (7.14.1.1).	CON37-C
136		A function with a variable number of arguments attempts to access its varying arguments other than through a properly declared and initialized <code>va_list</code> object, or before the <code>va_start</code> macro is invoked (7.16, 7.16.1.1, 7.16.1.4).	
137		The macro <code>va_arg</code> is invoked using the parameter <code>ap</code> that was passed to a function that invoked the macro <code>va_arg</code> with the same parameter (7.16).	
138		A macro definition of <code>va_start</code> , <code>va_arg</code> , <code>va_copy</code> , or <code>va_end</code> is suppressed in order to access an actual function, or the program defines an external identifier with the name <code>va_copy</code> or <code>va_end</code> (7.16.1).	MSC38-C
139		The <code>va_start</code> or <code>va_copy</code> macro is invoked without a corresponding invocation of the <code>va_end</code> macro in the same function, or vice versa (7.16.1, 7.16.1.2, 7.16.1.3, 7.16.1.4).	
140		The type parameter to the <code>va_arg</code> macro is not such that a pointer to an object of that type can be obtained simply by postfixing a <code>*</code> (7.16.1.1).	
141		The <code>va_arg</code> macro is invoked when there is no actual next argument, or with a specified type that is not compatible with the promoted type of the actual next argument, with certain exceptions (7.16.1.1).	DCL10-C
142		The <code>va_copy</code> or <code>va_start</code> macro is called to initialize a <code>va_list</code> that was previously initialized by either macro without an intervening invocation of the <code>va_end</code> macro for the same <code>va_list</code> (7.16.1.2, 7.16.1.4).	

UB	Class	Description	Guideline
----	-------	-------------	-----------







[SEI CERT C Coding Standard](#) / [4 Back Matter](#)

144		The member designator parameter of an <code>offsetof</code> macro is an invalid right operand of the <code>.</code> operator for the type parameter, or designates a bit-field (7.19).	
145		The argument in an instance of one of the integer-constant macros is not a decimal, octal, or hexadecimal constant, or it has a value that exceeds the limits for the corresponding type (7.20.4).	
146		A byte input/output function is applied to a wide-oriented stream, or a wide character input/output function is applied to a byte-oriented stream (7.21.2).	
147		Use is made of any portion of a file beyond the most recent wide character written to a wide-oriented stream (7.21.2).	
148		The value of a pointer to a <code>FILE</code> object is used after the associated file is closed (7.21.3).	FIO42-C , FIO46-C
149		The stream for the <code>fflush</code> function points to an input stream or to an update stream in which the most recent operation was input (7.21.5.2).	
150		The string pointed to by the mode argument in a call to the <code>fopen</code> function does not exactly match one of the specified character sequences (7.21.5.3).	
151		An output operation on an update stream is followed by an input operation without an intervening call to the <code>fflush</code> function or a file positioning function, or an input operation on an update stream is followed by an output operation with an intervening call to a file positioning function (7.21.5.3).	FIO39-C
152		An attempt is made to use the contents of the array that was supplied in a call to the <code>setvbuf</code> function (7.21.5.6).	
153		There are insufficient arguments for the format in a call to one of the formatted input/output functions, or an argument does not have an appropriate type (7.21.6.1, 7.21.6.2, 7.29.2.1, 7.29.2.2).	FIO47-C
154		The format in a call to one of the formatted input/output functions or to the <code>strftime</code> or <code>wcsftime</code> function is not a valid multibyte character sequence that begins and ends in its initial shift state (7.21.6.1, 7.21.6.2, 7.27.3.5, 7.229.2.1, 7.29.2.2, 7.29.5.1).	
155		In a call to one of the formatted output functions, a precision appears with a conversion specifier other than those described (7.21.6.1, 7.29.2.1).	FIO47-C
156		A conversion specification for a formatted output function uses an asterisk to denote an argument-supplied field width or precision, but the corresponding argument is not provided (7.21.6.1, 7.29.2.1).	
157		A conversion specification for a formatted output function uses a <code>#</code> or <code>0</code> flag with a conversion specifier other than those described (7.21.6.1, 7.29.2.1).	FIO47-C
158		A conversion specification for one of the formatted input/output functions uses a length modifier with a conversion specifier other than those described (7.21.6.1, 7.21.6.2, 7.29.2.1, 7.29.2.2).	FIO47-C
159		An <code>s</code> conversion specifier is encountered by one of the formatted output functions, and the argument is missing the null terminator (unless a precision is specified that does not require null termination) (7.21.6.1, 7.29.2.1).	





UB	Class	Description	Guideline
161		A % conversion specifier is encountered by one of the formatted input/output functions, but the complete conversion specification is not exactly %% (7.21.6.1, 7.21.6.2, 7.29.2.1, 7.29.2.2).	FIO47-C
162		An invalid conversion specification is found in the format for one of the formatted input/output functions, or the <code>strftime</code> or <code>wcsftime</code> function (7.21.6.1, 7.21.6.2, 7.27.3.5, 7.29.2.1, 7.29.2.2, 7.29.5.1).	FIO47-C
163		The number of characters or wide characters transmitted by a formatted output function (or written to an array, or that would have been written to an array) is greater than <code>INT_MAX</code> (7.21.6.1, 7.29.2.1).	
164		The number of input items assigned by a formatted input function is greater than <code>INT_MAX</code> (7.21.6.2, 7.29.2.2).	
165		The result of a conversion by one of the formatted input functions cannot be represented in the corresponding object, or the receiving object does not have an appropriate type (7.21.6.2, 7.29.2.2).	
166		A c, s, or [conversion specifier is encountered by one of the formatted input functions, and the array pointed to by the corresponding argument is not large enough to accept the input sequence (and a null terminator if the conversion specifier is s or []) (7.21.6.2, 7.29.2.2).	
167		A c, s, or [conversion specifier with an l qualifier is encountered by one of the formatted input functions, but the input is not a valid multibyte character sequence that begins in the initial shift state (7.21.6.2, 7.29.2.2).	
168		The input item for a %p conversion by one of the formatted input functions is not a value converted earlier during the same program execution (7.21.6.2, 7.29.2.2).	
169		The <code>vfprintf</code> , <code>vfscanf</code> , <code>vprintf</code> , <code>vscanf</code> , <code>vsnprintf</code> , <code>vsprintf</code> , <code>vsscanf</code> , <code>vfwprintf</code> , <code>vfwscanf</code> , <code>vswprintf</code> , <code>vswscanf</code> , <code>vwprintf</code> , or <code>vwscanf</code> function is called with an improperly initialized <code>va_list</code> argument, or the argument is used (other than in an invocation of <code>va_end</code>) after the function returns (7.21.6.8, 7.21.6.9, 7.21.6.10, 7.21.6.11, 7.21.6.12, 7.21.6.13, 7.21.6.14, 7.29.2.5, 7.29.2.6, 7.29.2.7, 7.29.2.8, 7.29.2.9, 7.29.2.10).	
170		The contents of the array supplied in a call to the <code>fgets</code> , <code>gets</code> , or <code>fgetws</code> function are used after a read error occurred (7.21.7.2, 7.21.7.7, 7.293.2).	FIO40-C
171		The file position indicator for a binary stream is used after a call to the <code>ungetc</code> function where its value was zero before the call (7.21.7.11).	
172		The file position indicator for a stream is used after an error occurred during a call to the <code>fread</code> or <code>fwrite</code> function (7.21.8.1, 7.21.8.2).	
173		A partial element read by a call to the <code>fread</code> function is used (7.21.8.1).	
174		The <code>fseek</code> function is called for a text stream with a nonzero offset and either the offset was not returned by a previous successful call to the <code>ftell</code> function on a stream associated with the same file or <code>whence</code> is not <code>SEEK_SET</code> (7.21.9.2).	
175		The <code>fsetpos</code> function is called to set a position that was not returned by a previous successful call to the <code>fgetpos</code> function on a stream associated with the same file (7.21.9.3).	
176		A non-null pointer returned by a call to the <code>calloc</code> , <code>malloc</code> , or <code>realloc</code> function with a zero requested size is used to access an object (7.22.3).	MEM04-C

UB	Class	Description	Guideline
		multiple of the alignment (7.22.3.1).	
179		The pointer argument to the <code>free</code> or <code>realloc</code> function does not match a pointer earlier returned by <code>calloc</code> , <code>malloc</code> , or <code>realloc</code> , or the space has been deallocated by a call to <code>free</code> or <code>realloc</code> (7.22.3.3, 7.22.3.5).	MEM34-C
180		The value of the object allocated by the <code>malloc</code> function is used (7.22.3.4).	
181		The values of any bytes in a new object allocated by the <code>realloc</code> function beyond the size of the old object are used (7.22.3.5).	EXP33-C
182		The program calls the <code>exit</code> or <code>quick_exit</code> function more than once, or calls both functions (7.22.4.4, 7.22.4.7).	ENV32-C , ERR04-C
183		During the call to a function registered with the <code>atexit</code> or <code>at_quick_exit</code> function, a call is made to the <code>longjmp</code> function that would terminate the call to the registered function (7.22.4.4, 7.22.4.7).	ENV32-C
184		The string set up by the <code>getenv</code> or <code>strerror</code> function is modified by the program (7.22.4.6, 7.24.6.2).	ENV30-C
185		A signal is raised while the <code>quick_exit</code> function is executing (7.22.4.7).	
186		A command is executed through the <code>system</code> function in a way that is documented as causing termination or some other form of undefined behavior (7.22.4.8).	
187		A searching or sorting utility function is called with an invalid pointer argument, even if the number of elements is zero (7.22.5).	
188		The comparison function called by a searching or sorting utility function alters the contents of the array being searched or sorted, or returns ordering values inconsistently (7.22.5).	
189		The array being searched by the <code>bsearch</code> function does not have its elements in proper order (7.22.5.1).	
190		The current conversion state is used by a multibyte/wide character conversion function after changing the <code>LC_CTYPE</code> category (7.22.7).	
191		A string or wide string utility function is instructed to access an array beyond the end of an object (7.24.1, 7.29.4).	
192		A string or wide string utility function is called with an invalid pointer argument, even if the length is zero (7.24.1, 7.29.4).	
193		The contents of the destination array are used after a call to the <code>strxfrm</code> , <code>strftime</code> , <code>wcsxfrm</code> , or <code>wcsftime</code> function in which the specified length was too small to hold the entire null-terminated result (7.24.4.5, 7.27.3.5, 7.29.4.4.4, 7.29.5.1).	
194		The first argument in the very first call to the <code>strtok</code> or <code>wcstok</code> is a null pointer (7.24.5.8, 7.29.4.5.7).	
195		The type of an argument to a type-generic macro is not compatible with the type of the corresponding parameter of the selected function (7.25).	

UB	Class	Description	Guideline

		is less than the year 1000 (7.27.3.1).	
198		The argument corresponding to an <code>s</code> specifier without an <code>l</code> qualifier in a call to the <code>fwprintf</code> function does not point to a valid multibyte character sequence that begins in the initial shift state (7.29.2.11).	
199		In a call to the <code>wcstok</code> function, the object pointed to by <code>ptr</code> does not have the value stored by the previous call for the same wide string (7.29.4.5.7).	
200		An <code>mbstate_t</code> object is used inappropriately (7.29.6).	EXP33-C
201		The value of an argument of type <code>wint_t</code> to a wide character classification or case mapping function is neither equal to the value of <code>WEOF</code> nor representable as a <code>wchar_t</code> (7.30.1).	
202		The <code>iswctype</code> function is called using a different <code>LC_CTYPE</code> category from the one in effect for the call to the <code>wctype</code> function that returned the description (7.30.2.2.1).	
203		The <code>towctrans</code> function is called using a different <code>LC_CTYPE</code> category from the one in effect for the call to the <code>wctrans</code> function that returned the description (7.30.3.2.1).	

Graphical symbols used in the preceding table:

Symbol	C11 Classification
	Critical Undefined Behavior
	Bounded Undefined Behavior
	Undefined Behavior (information/confirmation needed)
	Possible Conforming Language Extension