ISO/IEC/JTC 1/SC 22/WG 23 N0843

ISO/IEC JTC 1/SC 22 **N 5323**

ISO/IEC JTC 1/SC 22

Programming languages, their environments and system software interfaces

Secretariat: ANSI (United States)

Document type:	Summary of Voting/Table of Replies
Title:	Summary of Voting on PDTR 24772-1, Information technology -- Programming languages -- Guidance to avoiding vulnerabilities in programming languages -- Part 1: Language independent
Status:	The PDTR ballot received 100% approval with comments from Canada and Poland. The ballot results and the accompanying comments are forwarded to the next SC 22/WG 23 meeting for review and resolution of the comments and preparation of an approved disposition of comments document, a revised text, and a recommendation for further processing. The Project Editor is instructed to prepare the proposed disposition of comments document as soon as possible.
Date of document:	2018-11-25
Expected action:	INFO
Email of secretary:	mmiller@ansi.org
Committee URL:	https://isotc.iso.org/livelink/livelink/open/jtc1sc22

Result of voting

Ballot Information	
Ballot reference	ISO/IEC PDTR 24772-1 - JTC001-SC22-N5310
Ballot type	DTR
Ballot title	Information technology -- Programming languages -- Guidance to avoiding vulnerabilities in programming languages -- Part 1: Language independent
Opening date	2018-09-29
Closing date	2018-11-24
Note	

Member responses:	
Votes cast (21)	Austria (ASI) Bulgaria (BDS) Canada (SCC) China (SAC) Denmark (DS) Finland (SFS) France (AFNOR) Germany (DIN) Italy (UNI) Japan (JISC) Kazakhstan (KAZMEMST) Korea, Republic of (KATS) Netherlands (NEN) Poland (PKN) Russian Federation (GOST R) Slovenia (SIST) Spain (UNE) Switzerland (SNV) Ukraine (DSTU) United Kingdom (BSI) United States (ANSI)
Comments submitted (1)	Sweden (SIS)
Votes not cast (0)	

Questions:	
Q.1	"Do you approve the draft for publication?"

Votes by members	Q.1
Austria (ASI)	Abstention
Bulgaria (BDS)	Abstention

Canada (SCC)	Approval with comments
China (SAC)	Approval
Denmark (DS)	Abstention
Finland (SFS)	Abstention
France (AFNOR)	Abstention
Germany (DIN)	Abstention
Italy (UNI)	Approval
Japan (JISC)	Approval
Kazakhstan (KAZMEMST)	Approval
Korea, Republic of (KATS)	Approval
Netherlands (NEN)	Abstention
Poland (PKN)	Approval with comments
Russian Federation (GOST R)	Approval
Slovenia (SIST)	Abstention
Spain (UNE)	Abstention
Switzerland (SNV)	Abstention
Ukraine (DSTU)	Approval
United Kingdom (BSI)	Approval
United States (ANSI)	Approval

Answers to Q.1: "Do you approve the draft for publication?"		
9 x	**Approval**	**China (SAC)** **Italy (UNI)** **Japan (JISC)** **Kazakhstan (KAZMEMST)** **Korea, Republic of (KATS)** **Russian Federation (GOST R)** **Ukraine (DSTU)** **United Kingdom (BSI)** **United States (ANSI)**
2 x	**Approval with comments**	**Canada (SCC)** **Poland (PKN)**
0 x	**Disapproval**	
10 x	**Abstention**	**Austria (ASI)** **Bulgaria (BDS)** **Denmark (DS)** **Finland (SFS)** **France (AFNOR)**

Germany (DIN) **Netherlands (NEN)** **Slovenia (SIST)** **Spain (UNE)** **Switzerland (SNV)**

Comments from Voters		
Member:	Comment:	Date:
Canada (SCC)	***Comment File***	2018-10-26 18:29:02
CommentFiles/ISO_IEC PDTR 24772-1 - JTC001-SC22-N5310_SCC.doc		
Poland (PKN)	***Comment File***	2018-11-22 11:09:56
CommentFiles/ISO_IEC PDTR 24772-1 - JTC001-SC22-N5310_PKN.doc		

Comments from Commenters		
Member:	Comment:	Date:
Sweden (SIS)	***Comment***	2018-11-22 14:17:40
Abstention		

Template for comments and secretariat observations

Date:2018-11-25	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
CA 001		06.02.5		ed	"Avoid explicit type conversion of data values except when there is no alternative. Document such occurrences so that the justification is made available to maintainers." This could be read as encouraging implicit type conversions over explicit ones, which I believe is the opposite of the intent. Further, while I can possibly imagine it might make sense to document implicit conversions, I think the explicit conversion mostly reduces or eliminates the need to document the conversions. There is no mention in the previous bullet about documenting implicit conversions, where the need is greater.	Suggest removing "explicit" from this bullet, i.e. "Avoid [explicit] type conversion of data values except when there is no alternative. Document such occurrences so that the justification is made available to maintainers."	
CA 002		06.03.3		ed	"Problems can arise when programmers mix their techniques to reference the bits or output the bits. Problems can arise when programmers mix arithmetic and logical operations to reference the bits or output the bits. " There appears to be two versions of this sentence. Are they both needed?	Delete one of the two sentences if both are not needed. Otherwise, the second sentence should probably begin with "Problems can {also} arise ...."	
PL 003		06.04	06.4.5	te	Last subpoint: should be completed by "or use a suitable compensated summation algorithm" to read	When adding (or subtracting) sequences of numbers, sort and add (or subtract) them from smallest to largest in absolute value or use a suitable compensated summation algorithm to avoid loss of precision.	
CA 004		06.08.3		ed	"When an array has been allocated storage on the stack{,} an out-of-bounds write access may modify internal runtime housekeeping information (for example, a function's return address) which	Add missing comma	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

Template for comments and secretariat observations

Date:2018-11-25	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					might change a program's control flow"		
CA 005		06.14.06		te	"Language specifiers should design generics in such a way that any attempt to instantiate a generic with constructs that do not provide the required capabilities results in a compile-time error."	This seems completely unrelated to the vulnerability, nothing to do with free, but does sound like it could be applied to a different vulnerability relating to generics and templates. Is this out of place? Perhaps a cut and paste error?	
CA 006		06.17.04		te	"Languages that treat letter case as significant. Some languages do not differentiate between names with differing case, while others do." This suggests that the problem is with languages that treat case as significant, but the description in 6.17.1, in the 4th paragraph states; "There is also an issue where identifiers appear distinct to a human but identical to the computer" implies the problem is with languages where case is insignificant. In reality, I think confusion can come from either type of language. But mostly only in cases of languages where variables do not need to be declared.	Suggest replacing the 4th bullet with a more critical issue; "Languages that allow variables to be implicitly declared." An example that comes to mind here is BASIC, where, at least in earlier versions of the language, one could misspell the name of a variable and implicitly get a second variable, which might not be noticed by the programmer.	
CA 007		06.22.03		te	This vulnerability appears to cover elaboration errors? That is another source of such an error that is not described here, where a global variable can be used before initialization, even though in the code, the variable appears to be initialized. I do see it mentioned in 6.22.5.	This can be a very dangerous problem as the code looks correct. I think it would be helpful to describe this possibility better in 6.22.1 and 6.22.3	
CA 008		06.22.03		ed	"There is a special case [of]{for} pointers or access types."	Replace "of" with "for", or perhaps "associated with"	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

Template for comments and secretariat observations

Date:2018-11-25	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
CA 009	4th bullet	06.22.05		te	There is a recommendation to initializing variables during elaboration, but that is precisely what causes this problem in most languages such as C++. The recommendation to initialize during elaboration only is good advice for languages that can perform safe/orderly elaboration based on dependencies, so there should be clarification. A general recommendation for all languages that allow global declarations would be to avoid creating global objects. For example, dont create a singleton object if all that is needed is function calls that generate a result based on the input parameters. For languages that do not support safe elaboration, the recommendation should be to NOT initialize the object during elaboration, but instead either defer creation of the object until after elaboration, or protect access to the object by setting a flag indicating if the object has been initialized or not.	Clariify the advice is only recomnmended if the language supports safe elaboration, otherwise either try to defer creation of the object until after elaboration (such via access using a pointer), or by putting guards around the object to ensure that the first access either initializes the object, or indicates failure if the object is not yet initialized.	
CA 010		06.23.01		ed	"Each language provides rules of precedence and associativity, for each expression that operands bind to which operators. T"	I am unable to parse this sentence. Please reword.	
CA 011	6th bullet	06.25.05		ed	The 6th bullet says avoid null statements, but I think the intent there (expressions that do nothing or are not executed, for instance) is getting confused with language defined null statements, which is actually recommended in 6.25.6, 1st bullet.	Clarify that the 6th bullet is not referring to language defined null statements, but rather expressions or statements that do not have an effect, or something along those lines.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

Template for comments and secretariat observations

Date:2018-11-25	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
CA 012		06.25.06		ed	"Languages should consider providing warnings for statements that are unlikely to be right such as statements [without]{with} side effects. "	The logic of the sentence is backwards. “Without” should be replaced with “with”.	
CA 013		06.26.03		te	Another mechanism of failure, that could be mentioned is related to code bloat. A significant portion of dead code might effect the performance of the application (when loading the executable into memory, or perhaps cause linking problems by using up available code space on a tightly constrained embedded system.		
CA 014		06.26.05		ed	"When a developer identifies code that is dead because a conditional [consistently]{statically} evaluates "	Suggest replacing “consistently” with “statically”	
CA 015		06.34.04		te	Probably variadic functions should be given special mention, such as printf(), where a format string is passed as an argument, and the logic of library, rather than the compiler, assumes that the number of parameters passed matches the format string. A recommendation for such languages that support variadic functions, could be to use altnernate calls, such as streaming, where the compiler can validate the correct parameters are passed, rather than relying on runtime functions.	Suggest mentioning variadic functions, and recommend avoiding their use if possible.	
CA 016		06.39.04		ed	"Languages {that} reclaim memory under programmer control can exhibit heap fragmentation and memory leaks."	Add “that”	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

Date:2018-11-25	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
CA 017		06.39.06		ed	The word "pragma" is wrong font and size		
CA 018		06.40.03		ed	" about the types it can legally be instantiated with." Avoid ending a sentence with a preposition.	Suggest replacing with => "about the parameterized types."	
CA 019		06.40.03	Paragraph 3	ed	There are two "For Example"s, which looks and reads as though one example was meant to replace the other.	Remove one of the examples, if only 1 is needed, or preface the second with something like, “As a second example,”	
CA 020		06.42.03		ed	"of a class which [re]dispatches to the implementation" I dont think this applies only to redispatching, also to ordinary dispatching	Suggest generalizing and removing "re"	
CA 021		06.50.06		te	"Provide a mechanism to determine which exceptions might be thrown by a called library routine." It is not clear this is good advice. Based on experience in other languages (in particular Java and C++), exception "signatures" have not generally provided the value expected for them. Here is an example paper describing some of the issues in the context of Java: http://literatejava.com/exceptions/checked-exceptions-javas-biggest-mistake/ <http://literatejava.com/exceptions/checked-exceptions-javas-biggest-mistake/>	Consider whether this sentence should be removed or not. On the other hand, this is only advice, which the reader can ignore. There may be some who feel this is still good advice. Or perhaps clarification would be helpful to mention possible pitfalls of following this advice.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

Template for comments and secretariat observations

Date:2018-11-25	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					Here is a discussion of the problems with C++ exception specifications: http://www.gotw.ca/publications/mill22.htm <http://www.gotw.ca/publications/mill22.htm>		
PL 022		Gereneral		ge	the document even though claiming to be language independent mostly depends on how Ada/C languages are implemented and mostly describes just them. Paper should either stay totally language independant and describe general concepts like floating point arithmetics problems etc. or only papers for specific languages should remain.	Leave only totally language independent topics in the document.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

Date:2018-11-25	Document:	Project:

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat

ISO_IEC PDTR 24772-1 - JTC001-SC22-N5310_PKN.doc: Collation successful

ISO_IEC PDTR 24772-1 - JTC001-SC22-N5310_SCC.doc: Collation successful

Collation of files was successful. Number of collated files: 2

SELECTED (number of files): 2

PASSED TEST (number of files conformed to CCT table model): 2

FAILED TEST (number of files conformed to CCT table model): 0

CCT - Version 2018.2

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial