ISO/IEC JTC 1/SC 22 N 5321

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 24772-3, Information technology -- Programming languages -- Guidance to avoiding vulnerabilities in programming languages -- Part 3: C

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 24772-3 - JTC001-SC22-N5312
Ballot type	DTR
Ballot title	Information technology -- Programming languages -- Guidance to avoiding vulnerabilities in programming languages -- Part 3: C
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***	2018-11-22 14:17:48
CommentFiles/ISO_IEC PDTR 24772-3 - JTC001-SC22-N5312_SCC.doc		
Poland (PKN)	***Comment File***	2018-11-22 10:38:55
CommentFiles/ISO_IEC PDTR 24772-3 - JTC001-SC22-N5312_PKN.doc		

Comments from Commenters		
Member:	Comment:	Date:
Sweden (SIS)	***Comment***	2018-11-22 14:18:16
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
PL 001		03.01		ge	Rather than copy-pasting whole section from the C standard maybe it's just better to reference it		
PL 002		04		te	Actually according to the C standard there is difference in the semantics: (based on N1570) 6.2.5 Types: 20. A pointer type describes an object whose value provides a reference to an entity of the referenced type. It means. That pointer is a type, while reference is the pointer's value property	“Unlike some other languages, in C the terms ‘pass by reference’, ‘pass by pointer’, ‘pass by address’ have the same meaning”	
PL 003		05	Table, point 8	te	should actually also mention, that it can be intended to wrap-around the result, which is also fine and this is why you can do it on unsigned integers without UB	Check, that the result of an operation on an unsigned integer value will not cause wrapping unless it can be shown that wrapping cannot occur or it’s intended behavior.	
PL 004		06.02.2	Point 2	te	Being aware of the promotion/conversion rules won't prevent from actually doing mistakes with them as they are easy to be skipped.	“Enable all compiler warnings regarding implicit conversions or use static analysis tools to show the warning”	
PL 005	1	06.03.1		ed	"C supports a variety of sized for integers such as ... . Each may either be signed or unsigned". As to my understanding this means, that it can be signed and unsigned as well.	C supports a variety of sized for signed integers ... Each signed integer has it's unsigned counterpart.	
PL 006	2	06.08.1		ed	“unspecified”	“undefined”	
PL 007		06.12.02	Point 2	te	1. To all my knowledge the pointer arithmetic is there actually to prevent you from doing the calculation error. 2. index operator do just the same thing as we can manually with pointer arithmetic.	Remove the point	
PL 008		06.14.02		te	If the document recommends to malloc with the macro that does automatic casts, then most	Use macro to free and set pointer to NULL example macro definition:	

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
					probably it should recommend freeing memory with macro, that automatically sets the pointer to the null value	#define freeAndNull(P) free((P));(P)=NULL; (or any similar macro)	
PL 009		06.26.02		te	using // instead of /**/ - I do not see how this can help. Especially with the given example. With /**/ You exactly see where the comment ends + even the simplest editors colors the comments differently than code, so it's hard to miss that	Remove the guidance	
PL 010		06.29.02		te	Do not modify a loop control variable within a loop. Usefulness of this holds true for "for" loops. This guidance is more than misleading in case of while or do while loops.	Do not modify a loop control variable within a "for" loop	
CA 011		06.32.02		ed	Typo	"Follow the g uidance contained in ...." -> "Follow the guidance contained in ...."	
PL 012		06.39.02		te	If I understand correctly paper proposes to use garbage collectors. This is in my opinion wrong, as a) this is not supported by the C language standard (language level). Even if some compiler/lib supports some kind of garbage collecting for C it will be not standard C code. I am not sure it's good idea to recommend non standard approaches towards coding guidelines. Instead document should propose how to use language features in such a way, that it's safe b) using garbage collector will cause other kinds of vulnerabilities especially with real-time applications (as it's noticed in the document)	Remove the guidance	
PL 013	1	06.46.01		te	Parameter passing is either pass by reference or pass by value. This is wrong. It's always pass by value.	Remove first sentence in 6.46.1	

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
PL 014		06.51.02		te	I would add one guidance in here. Apply coding standards so that macros are easily differentiable from the inline function	→ Use coding guidelines to make function-like macros differentiable from inline functions e.g. function-like macros should be named with capital letters	
PL 015		06.54		ed	The section is too generic.	Leaving only first point from 6.54.2 would suffice	
CA 016		06.56.02		ed	Typo	"Follow the g uidance contained in ...." -> "Follow the guidance contained in ...."	

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

ISO_IEC PDTR 24772-3 - JTC001-SC22-N5312_PKN.doc: Collation successful

ISO_IEC PDTR 24772-3 - JTC001-SC22-N5312_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