ISO/IEC JTC1/SC22
Languages
Secretariat: CANADA (SCC)

ISO/IEC JTC1/SC22

N673

AUGUST 1989

TITLE:	Summary of Voting and Comments received on : DP9899-Programming Language C
SOURCE:	Secretariat ISO/IEC JTC1/SC22
WORK ITEM:	JTC1.22.20
STATUS:	New
CROSS REFERENCE:	N619
DOCUMENT TYPE:	Summary of Voting/Comments on DP
ACTION:	For information to SC22 Member Bodies. The comments have submitted to SC22/WG14 for recommendation on further processing of DP9899.

Address reply to: ISO/IEC JTC1/SC22 Secretariat
J.L. Côté, 140 O'Connor St., 10th Floor
Ottawa, Ont., Canada K1A 0R5
Telephone: (613)957-2496   Telex: 053-3336

<u>SUMMARY OF VOTING ON</u>:

Letter Ballot Reference No:	SC22 N619
Circulated by	:JTC1/SC22
Circulation Date	:1989-03-03
Closing Date	:1989-06-03

<u>SUBJECT</u>:DP9899: Information Processing Systems-
Programming Language C

The following responses have been received:

'P' Members supporting the proposal , without comments	: 07(Austria,Canada,Czechoslovakia, Finland,Netherlands,New Zealand, USSR)
'P' Members supporting the proposal, with comments	: 04 (Denmark,France, Japan,UK)
'P' Members not supporting the proposal:	00
'P' Members abstaining	:00
'P' Members not voting	:09(Belgium,China,Germany FR, Hungary,Iran,Italy,Sweden, Switzerland,USA)

<u>Comments</u>:

Attachment 1 - Denmark
Attachment 2 - France
Attachment 3 - Japan
Attachment 4 - UK

<u>Secretariat Action</u>:
The SC22 Secretariat will forward the attached comments to WG14 for consideration and recommendation on further processing of DP9899.

ISO/IEC JTC1/SC22 LETTER BALLOT SUMMARY

PROJECT NO: JTC1.22.20

SUBJECT: DP9899:Information Processing Systems-
Programming Language C

Reference Document No: DP9899 **Ballot Document No:** N619

Circulation Date: 1989-03-03 **Closing Date:** 1989-06-30

Circulated To: SC22 P,O,L **Circulated By:** Secretariat

SUMMARY OF VOTING AND COMMENTS RECEIVED

'P' Members	Approve	Disapprove	Abstain	Comments	Not Voting
Austria	(x)	( )	( )	( )	( )
Belgium	( )	( )	( )	( )	( )
Canada	(x)	( )	( )	( )	( )
China	( )	( )	( )	( )	( )
Czechoslovakia	(x)	( )	( )	( )	( )
Denmark	(x)	( )	( )	(x)	( )
Finland	(x)	( )	( )	( )	( )
France	(x)	( )	( )	(x)	( )
Germany F.R.	( )	( )	( )	( )	( )
Hungary	( )	( )	( )	( )	( )
Iran	( )	( )	( )	( )	( )
Italy	( )	( )	( )	( )	( )
Japan	(x)	( )	( )	(x)	( )
Netherlands	(x)	( )	( )	( )	( )
New Zealand	(x)	( )	( )	( )	( )
Sweden	( )	( )	( )	( )	( )
Switzerland	( )	( )	( )	( )	( )
UK	(x)	( )	( )	(x)	( )
USA	( )	( )	( )	( )	( )
USSR	(x)	( )	( )	( )	( )
'O' Members					
Australia	( )	( )	( )	( )	( )
Brazil	( )	( )	( )	( )	( )
German Dem Rep.	( )	( )	( )	( )	( )
Iceland	( )	( )	( )	( )	( )
India	( )	( )	( )	( )	( )
Korea	( )	( )	( )	( )	( )
Norway	( )	( )	( )	( )	( )
Poland	( )	( )	( )	( )	( )
Portugal	( )	( )	( )	( )	( )
Singapore	( )	( )	( )	( )	( )
Turkey	( )	( )	( )	( )	( )
Thailand	( )	( )	( )	( )	( )
Yugoslavia	( )	( )	( )	( )	( )

Attachment 1 - Denmark comments is missing attachment I

AFNOR'S COMMENTS RELATING TO SECOND DP 9899

PROGRAMMING LANGUAGE "C"

(JTC1/SC22 N 619)

AFNOR approves second DP 9899 (doc JTC1/SC22 N 619) with the following remarks.

1 - Relating to the DP itself, we are pleased to say that our remarks made at the first DP stage have been taken into account and fairly solved.

2 - We note with satisfaction that the Rationale is enclosed in the present Ballot. We think that it might be useful that the document be part of the final standard.

89/64809

Attachment 3 - Japan comments is missing

Subject: ISO Document DP9899 - N619
(Second DP9899 - Programming Language C)

89/63401.

Based on a review of the subject document, the UK votes YES though with considerable misgivings, overcome only by accepting that there is a general wish among the 'C' community for a standard to be published as soon as possible, even if it has deficiencies. An excessive number of undefined aspects and ambiguities remain in the document and those identified in the UK are listed in an attached document. Our yes vote is on the understanding that WG14 will proceed immediately to produce an ancillary document addressing such issues not already resolved, by the time the primary document reaches DIS stage, and that this document, once agreed also be adopted by the US member body for its domestic standard and by any other member body with equivalent domestic standards.

The UK believes that the first revision of the 'C' standard, incorporating the interpretations in the additional document just referred to, should be developed by WG14 directly and that the terms of reference of WG14 (Paris resolution) should be revised accordingly.

Comments on the December 7, 1988 Draft Proposed C standard

1) page 2 line 36, page 3 line 6. Definition of Byte and Character are self referencing and circular.

2) page 2-3. Definitions should be in alphabetic order.

3) page 3 line 30. Paragraph should be given its own heading.

4) page 3 line 32. "... or by the omission ...", but this is "Unspecified behaviour" as defined on line 21 above.

5) page 6 line 37. "Preporcessing directives are executes and macro invocations are expanded"

page 87 line 1-5. "A preprocessing directive ... and is ended by the next new-line character"

page 90 line 40. "within the sequence of preprocessing tokens ... new-line is considered a normal white-space character"

These three statements are not sufficient to define the following:

```
#define f(a,b) a+b
#if f(1,
      2)
...
```

It should be defined whether the preprocessing directive rule or macro expansion wins.

6) page 8. It should be a constraint error or explictly undefined if no function called main exists in the executable.

7) page 21 Implementation limits. It is currently undefined behaviour if the implementation does not treat 31 characters as being significant. It should be a constraint error if the implementation does not support this feature.

8) page 21 line 26. It is undefined behaviour if label names are not unique within a function. This should be an error.

9) page 32 line 37. "... [], () and {} shall occur in pairs ...". They need only occur in pairs after translation phase 4.

10) page 35 line 13. A nit pick really.

in:

```
        int i;
        enum foo{a=1} x;
        i=(0,x) +1;    /* illegal */
        i=x+1     ;    /* legal */
```

The type of (0,x) is enum foo. However, it is not an object and the draft standard does not explicitly allow it to occur in an int context.

Does the committee intend this behaviour?

12) page 38. ".. with the value 0", should refer to the NULL macro.

13) page 39 footnote 33. Hang over from noalias days?

14) sizeof has type size_t. It should be a constraint error or explicitly undefined if the definition of size_t obtained from an include file does not match what the compiler considers the type of sizeof to be (the compiler has to maintain an internal type for sizeof in case the user does not include a header that defines this type).

The above comments also apply to ptrdiff_t, page 48 line 34.

15) An inferior implementation may flag a statement as being in error, while a quality implementation treats it as being legal. Consider:

```
char *p;
p=1-1;
```

A quality implementation would fold 1-1 to give zero. If the NULL macro is defined as 0 then the assignment is legal, it is the null-pointer-constant.

An inferior implementation would not fold the 1-1 and would flag the assignment as being in error.

The standard should explictly state that in an expression context only unfolded constants may become null-pointer-constant.

16) page 41 line 30. Suggest change of "... in the innermost block containing ..." be changed to "... in the innermost block, after any explicit declarations, containing ...". This extra wording keeps:

```
int (*g)()=f;
...
f();
```

illegal.

Without it the implicit declaration that occur on encountering the function call f() might make the initialiser given on the declaration of g legal. A one pass compiler would already have flagged the initialiser on g as being in error.

An even better solution would be to implicitly declare f with file scope.

17) page 53 line 13. "... or differently qualified versions of a compatible type, the result has the composite type; ...", but no rules are given on page 26 for creating a composite type for the expression. In:

```
x ? (const int *)y : (int *) z
```

what is the composite type of (const int *) and (int *) ?

also in:

```
x ? (const int *)y : (const void *) z
```

the result type is (void *), contradicting line 11 above.

18) page 58 line 16. This constraint currently allows zero sized structs, as in:

```
struct { struct a{int b;};}
```

The reference to "... no named members, ..." in page 61 line 25 presumably refers to bit fields.

19) page 68. What is the type of constant-expression in array declarations. In the following what are the array sizes?

```
int a1[32767+1];
int a2[65000*65000];
```

20) page 68 clause 3.5.4.3. An ambiguity in needs resolving in the parsing of the following:

a) `int x(T (U));`

b) `int x(T (U (int a, char b)));`

In (a) U is the type of the parameter to a function returning type T. From page 69 line 2: "In a paramter declaration, a single typedef name in parentheses is taken to be an abstract declarator that specifies a function with a single parameter, not as redundant parentheses around the identifier for a declarator."

Thus in the case of (b):

1) U could be a redundantly parenthesized name of a function which takes a parameter-type-list and returns type T, or

2) U could be the type returned by a function which takes a parameter-type-list; which in turn is the single parameter of a function returning type T.

21) page 71 line 5. It is undefined whether the following is legal or illegal:

```
typedef t[];
t a={1,2}, b{3,4,5};
```

The behaviour for this case should be defined.

22) page 72 line 43. Change "... its value is indeterminate." to "... its value is undefined.".

23) page 81 line 24. Change "... whose return type is void" to "... whose unqualified return type is void".

24) page 83 line 34-35. The fact that the storage-class specifier and type-specifier may be omitted, defaulting to extern and int respectively should be in the semantics clause, not an example.

25) page 84 line 1. The fact that parameter declarations may be omitted and default to int ought to be in the semantics section.

26) pointer to multi-dimensional arrays. Neils query.

28) page 82 line 22. Implies that there may be more than one external definition provided the object does not occur in an expression.

It should be a constraint or explictly undefined if this situation occurs.

29) There is no semantics given for the replacement oj object macros. This should be defined.

30) page 90. In:

```
#define f(a) a*g
#define g(a) f(a)
f(2)(9)
```

it should be defined whether this results in 2*f(9) or 2*9*g

31) page 168 line 36. "... first call ..." should read "... all calls ..."

SRB/DJ/srb
29 June 1989

89/634C