485   056
Document Number: WG14 N~~477~~/X3J11 95-~~033~~

C9X Revision Proposal
=====================

Title: Provide bool, true, and false, optionally as keywords
Author: Thomas Plum
Author Affiliation: Plum Hall Inc
Postal Address: PO Box 44610, Kanuela HI 96743 USA
E-mail Address: plum@plumhall.com
Telephone Number: +1 808 882 1255
Fax Number: +1 808 882 1556
Sponsor: ____
Date: 1995-06-17
Proposal Category:
   X New feature
Area of Standard Affected:
   X Language
Prior Art: C++
Target Audience: all
Related Documents (if any): C++ Draft Standard
Proposal Attached: X Yes
Proposal:

The C++ CD 14882 says, in 3.9.1 Fundamental Types, ...

8 Values of type bool are either true or false.26) There are no signed, unsigned, short, or long bool types or values. As described below, bool values behave as integral types. Values of type bool participate in integral promotions (4.5, 5.2.3). Although values of type bool generally behave as signed integers, for example by promoting (4.5) to int instead of unsigned int, a bool value can successfully be stored in a bit-field of any (nonzero) size.

Two options are possible:

#1) bool, true, and false are keywords in C9X.

#2) At the implementer's choice, they are either keywords or synonyms for types and values that are added to (presumably) <stddef.h> .

If option #2, a conforming C9X program would not depend upon whether they are keywords, typedef, macro, or enum. In my opinion, that is preferable, because that describes the uncertainty that already exists today.

Furthermore, there should be a standardized macro name, such as `__bool_true_false_are_defined` which indicates that the implementation provides definitions (either by option #1 or #2). This standardized "feature test" is very important, because there will be several years during which some implementations (at least of C++) will define the names, while other implementations will not. Headers for general distribution need to know whether or not to define their own bool, true, and false, and there needs to be a standardized way to ask. (I intend to propose the same requirement for C++.)