Doc Numbers: WG21/N0737
X3J16/95-0137
Date: 12-July-1995
Reply To: Jim Thomas
jim_thomas@taligent.com
Or: Yinsun Feng
yfeng@taligent.com
Taligent, Inc.
10201 N. DeAnza Blvd.
Cupertino, CA 95014

Annex Proposal: Floating-Point Notes

This is a proposal for an informative annex, entitled Floating-Point Notes, to the C++ draft standard.

Abstract

Like the C standard, the C++ draft standard allows ample latitude for floating-point arithmetic. It leaves much to "quality of implementation". Such loose specification can be problematic for programmers, especially considering portability. At the same time, C++ extensibility offers new opportunities for numerical programming, and numerical facilities are well represented in the C++ draft standard libraries. Implementors with a priority on supporting numerical computation will be faced with a number of floating-point issues, which though unimportant for C++ conformance, matter greatly to numerical programmers and users. Two particularly noteworthy areas are:

1. Implementation decisions that affect basic floating-point predictability—for all implementations with significant interest in numerical programming.

2. Implementation decisions related to support of IEEE standard floating-point (IEC 559, aka ANSI/IEEE 754, in essentially all modern hardware)—for all implementations with `is_iec559` (in `numeric_limits`) equal to `true`.

This annex notes issues and offers guidance to C++ implementors regarding treatment of floating point. The first part covers predictability, the second part covers IEEE support. (This annex borrows from "Floating-Point C Extensions", in the ANSI C committee's Technical Report on Numerical C Extensions, the result of a five year effort to address these issues for ISO C.)

Tentative Outline

Part 1—Predictability

- Expression evaluation. Regularize three common basic methods, providing at least one well suited method for essentially any floating-point hardware.
- Binary-decimal conversion. Define a useful, practical notion of correct rounding.
- Constant evaluation. Urge translation-time/execution-time consistency. Urge consistency with expression evaluation.
- Optimization. Note "optimizations" that change numerical values. Suggest a policy for use of hardware operations that implement multiple C++ operators as one.
- Documentation. Recommend what to provide.

Part 2—IEEE (IEC 559) support

- Type binding. Map IEEE types to C++ types.
- Operation binding. Map IEEE operations to C++ operators and functions.
- Function binding. Provide a C++ interface for IEEE recommended functions.
- Special values. Address issues about handling IEEE infinities, NaNs, and -0.
- Floating-point environment management. Note implementation prerequisites for making the global floating-point exception flags and rounding direction modes available to users.
- Optimization. Note "optimizations" that subvert use of special values, flags, and modes. Recommend a way to assure availability of IEEE features, yet retain important optimizations.
- Documentation. Recommend what to provide.