Doc. No.: WG21/N0962=X3J16/96-0144
Date: July 8, 1996
Project: C++ Standard Library
Reply to: David Vandevoorde (vandevod@cs.rpi.edu)
Philippe Le Mouel (philippe@roguewave.com)

High-performance C++ implementations for valarrays (Rev. 3)

Introduction

This proposal presents an alternative to version 2 which allows expression templates while allowing previously conformant implementations to remain so without requiring any changes.

Proposal

a) Renumber paragraph 26.3/3 to 26.3/6.
b) Insert the following 3 paragraphs (26.3/3-5):

3 Any function returning a valarray<T> is permitted to return an object of another type, provided all the const member functions of valarray<T> are also applicable to this type. This return type shall not add more than two levels of template nesting over the most deeply nested argument type[1].

4 Implementations introducing such replacement types shall provide additional functions and operators as follows:

- for every function taking a const valarray<T>&, identical functions taking the replacement types must be added;
- for every function taking two const valarray<T>& arguments, identical functions taking any combination of valarray<T> const& and replacement types must be added.

5 In particular, an implementation must allow a valarray<T> to be constructed from such replacement types and must allow assignments and computed assignments of such types to valarray<T>, slice_array<T>, gslice_array<T>, mask_array<T> and indirect_array<T>.

[1] Appendix B recommends a minimum number of recursively nested template instantiations. This requirement thus indirectly suggests a minimum allowable complexity for valarray expressions.