N4356 : Relaxed Array Type Declarator

Authors:

Carter Edwards hcedwar@sandia.gov

Christian Trott crtrott@sandia.gov

Related papers:

N4177 Multidimensional bounds, index and array_view, revision 4

N4222 Minimal Additions to the Array View Library for Performance and Interoperability

N4355 Shared Multidimensional Array with Polymorphic Layout

1 Scope

This paper proposes a relaxation of the array type declarator constraint specified in **8.3.4 Arrays, paragraph 3**. This relaxation is proposed to improve clarity, conciseness, and performance of the Shared Multidimensional Array with Polymorphic Layout proposal, N4355.

2 Current Constraint on Array Declarator

Paragraph 8.3.4.p3 states the following constraint on an array declaratory.

When several "array of" specifications are adjacent, a multidimensional array is created; only the first of the constant expressions that specify the bounds of the arrays may be omitted.

Thus in a multidimensional array object declarator of the form

```
D1 [ N0 ][ N1 ][ N2 ]...
```

only the first dimension **N0** may be omitted. The omitted dimension is either taken from an earlier declaration or mapped to pointer offset-dereference semantics.

3 Relaxed Constraint for Array Type Declarator

The current array declarator constraint is suitable for explicit declaration of a conventional, built-in array object. However, this constraint is not necessary in the declaration of a multidimensional array **type** that is not explicitly used to declare a conventional, built-in array object. Furthermore, relaxing this constraint for multidimensional array **type** declaration will enable clear and concise type specification for the proposed shared multidimensional array with polymorphic layout (N4355).

3.1 Proposal

The current Paragraph 8.3.4.p3 constraint is relaxed to the following.

When several "array of" specifications are adjacent to form a multidimensional array type specification only the first of the sequence of array bound constant expressions may be omitted for types used in the explicit declaration of a multidimensional array; otherwise any or all of the array bound constant expressions may be omitted.

3.2 Motivation

This change allows a multidimensional array **type** specification may have the form:

T[N0$_{opt}$][N1$_{opt}$][N2$_{opt}$]...

where **T** is possibly cv-qualified object type of array members, the count of **[N#$_{opt}$]** expressions is the rank of the multidimensional array, and each **N#$_{opt}$** is an optional integral constant expression denoting an explicitly declared array dimension.

In the shared multidimensional array with polymorphic layout proposal each omitted array bound (a.k.a., array dimension) implies an array bound that is specified at runtime. This allows a shared multidimensional array with polymorphic layout to merge explicitly and implicitly specified array bounds and enables optimization of the array multi-index offset computation when array bounds can be explicitly specified. The desired syntax for such a library class, proposed in N4355 and referenced here to elaborate this motivation, is as follows.

std::shared_array< T[N0$_{opt}$][N1$_{opt}$][N2$_{opt}$]... , Layout$_{opt}$ >

4 Derived Requirements

The array type property queries (20.10.5) **std::rank** and **std::extent** implementations must support multidimensional array type declarations with relaxed array bound constraint. It is expected that an implementation of the shared multidimensional array with polymorphic layout will utilize these property queries. The specification of these property queries does not need to be revised. However, the examples should be extended for clarity.

20.10.5.p2, add

```
assert(rank<int[][]>::value == 2);
```

20.10.5.p3, add

```
assert(extent<int[][],1>::value == 0);
```