`mdspan::size_type` should be `index_type`

Document number: P2599R0
Date: 2022-06-07
Project: Programming Language C++, Library Evolution Working Group
Reply-to: Nevin “☺” Liber, nliber@anl.gov

Table of Contents

Introduction

With the adoption of P2553R1, `mdspan::size_type` may now be a signed type. `size_type` is no longer an appropriate name for this type and it should be changed to `index_type`.

Motivation and Scope

Throughout the C++ standard, `size_type` stands for an unsigned type. `mdspan` and its related class templates should be consistent with this.

When P2553R0 was proposed, `extents::size_type` was going to be constrained to `unsigned_integral`. At the request of LEWG, that constraint was removed in P2553R1 and adopted via electronic polling.

Now that it can be a signed type, `size_type` is no longer the correct name for this. It should revert back to `index_type`, which was used in `mdspan` until P0009R11 when the following change was made:

Change all the sizes
from `ptrdiff_t` to `size_t` and `index_type` to `size_type`, for consistency with `span` and the rest of the standard library

In addition to `extents`, there are other class templates which take `Extents` as a template parameter and adopt the `size_type` typedef from `Extents` into their interface. Those class templates should also have their `size_type` typedefs changed to `index_type`.

Specifically, the following class templates should replace their usage of `size_type` with `index_type`:

- `extents`
- `layout_left::mapping`
- `layout_right::mapping`
- `layout_stride::mapping`
- `mdspan`

Impact On the Standard

Given that `mdspan` and its related classes are new class templates for C++23, the impact should be minimal. This should be applied to P0009 and P2553 (if that is still under LWG review) or the C++WD (if `mdspan` has already been adopted into the IS). Also, no feature test macro should be necessary.

Technical Specifications

The only normative changes proposed here are in the spellings of `size_type` to `index_type`, `SizeT` / `SizeType` to `IndexType`, `OtherSizeT` / `OtherSizeType` to `OtherIndexType` and `SizeTypes` to `IndexTypes`. No other normative wording changes are being proposed.

Both P0009 and P2553 are currently undergoing revisions as requested by LWG. If this proposal is approved, the author will apply these spelling changes to both those documents.

The drafts for these spelling changes can be found under https://github.com/nliber/cpp-proposals-pub/tree/P2553-P0009-size_type-to-index_type, based on the drafts found under https://github.com/mhoemmen/cpp-proposals-pub/tree/P2553-P0009-LWG-small-group-20220531.

Acknowledgements

This was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nation's exascale computing imperative. Additionally, this research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

References

P0009 mdspan, Christian Trott *et al*.
P2553 Make mdspan size_type controllable, Christian Trott *et al*.