Doc. no.: P0809R0
Date: 2017-10-12
Reply to: Titus Winters (titus@google.com),
Audience: LEWG/LWG

Comparing Unordered Containers

Abstract

Resolve issue 2831 by applying the proposed resolution. Comparing equality among unordered containers does not require identical hasher behavior, only identical comparison (Pred) behavior.

Background

The current wording on requirements for comparison of unordered containers says this [unord.req]:

> Two unordered containers `a` and `b` compare equal if `a.size() == b.size()` and, for every equivalent-key group `[Ea1, Ea2)` obtained from `a.equal_range(Ea1)`, there exists an equivalent-key group `[Eb1, Eb2)` obtained from `b.equal_range(Ea1)`, such that `is_permutation(Ea1, Ea2, Eb1, Eb2)` returns `true`.
> …
> The behavior of a program that uses `operator==` or `operator!=` on unordered containers is undefined unless the `Hash` and `Pred` function objects respectively have the same behavior for both containers and the equality comparison function for `Key` is a refinement of the partition into equivalent-key groups produced by `Pred`.

Notice that Pred is implicated in the equality definition, but Hash is not. Thus, the UB definition for heterogenous containers should not apply merely because of inequity among hashers - and in practice, this may be valuable because of hash seeding and randomization. Hash equality may be necessary for efficiency (a particularly poor hash function may cause the equal_range operations above to be linear in the size of the container), but not for correctness.

Proposed Wording

Change [unord.req]/p12 as indicated:

Two unordered containers a and b compare equal if a.size() == b.size() and, for every equivalent-key group [Ea1, Ea2) obtained from a.equal_range(Ea1), there exists an equivalent-key group [Eb1, Eb2) obtained from b.equal_range(Ea1), such that is_permutation(Ea1, Ea2, Eb1, Eb2) returns true. For unordered_set andunordered_map, the complexity of operator== (i.e., the number of calls to the == operator of the value_type, to the predicate returned by key_eq(), and to the hasher returned by hash_function()) is proportional to N in the average case and to N^2 in the worst case, where N is a.size(). For unordered_multiset and unordered_multimap, the complexity of operator== is proportional to $\sum E_i^2$ in the average case and to N^2 in the worst case, where N is a.size(), and E_i is the size of the ith equivalent-key group in a. However, if the respective elements of each corresponding pair of equivalent-key groups Ea_i and Eb_i are arranged in the same order (as is commonly the case, e.g., if a and b are unmodified copies of the same container), then the average-case complexity for unordered_multiset and unordered_multimap becomes proportional to N (but worst-case complexity remains $\mathcal{O}(N^2)$, e.g., for a pathologically bad hash function). The behavior of a program that uses operator== or operator!= on unordered containers is undefined unless the ~~Hash and~~ Pred function object~~s respectively have~~has the same behavior for both containers and the equality comparison operator for Key is a refinement of the partition into equivalent-key groups produced by Pred.