Doc. No.: J16/97-0076 == WG21/N1114
Date: September 30, 1997
Project: Programming Language C++
Reply to: Dave Abrahams <abrahams@motu.com>
Greg Colvin <greg@imrgold.com>

Making the C++ Standard Library More Exception Safe

In London we made some changes to the library to provide one exception safe container (*list*) and one exception safe function (*swap*). This paper identifies the few changes remaining to provide the full degree of exception safety proposed in J16/97-0048R1 == WG21/N1086R1. We remain convinced that without these changes the library does not yet specify sufficient exception safety.

One sticking point in London was the lack of a clear and simple statement of just what degree of safety the library should provide. A slightly over-simplified statement of the post-London status quo is:

- *swap()* is safe;
- *list* is safe.

To be more precise: *swap()* has no effects if it throws an exception, erasing elements off a *list* will not throw, and inserting elements on a *list* has no effects if it does throw. To be even more precise: *swap()* is safe on an associative container only if its *Compare* object can be copied without throwing.

We propose to enlarge that statement to:

- *swap* is safe;
- *list*, *map*, *set*, *multimap*, and *multiset* are safe;
- *vector* and *deque* are safe if they contain PODs;
- *stack* and *queue* are safe;
- iterators returned from standard containers are safe.

To be more precise: multiple-element insertions are safe only for *list*; an *insert()* or *erase()* on a *vector* or *deque* is safe only if it contains a type which can be copied without throwing; *pop_back()* and *pop_front()* will not throw; *push_back()* and *push_front()* have no effects if they do throw; and an iterator returned from a standard container can be copied without throwing.

The associative containers can be made safe almost easily as can *list*, and for the same reason: they are node-based, so any failure to construct a node simply leaves the container as it was, and given that destructors don't throw there is no reason for removing a node to fail. However, for multiple-element operations the need to keep elements sorted makes full recovery from a throw impractical.

Vector and *deque* are array-based containers, and so cannot fully recover when a contained type's copy operation throws, because of the need to copy elements when resizing or inserting. Objects like PODs, whose copy operations do not throw, are common enough that the more limited guarantee is still useful. However, push and pop operations do not require copying existing elements, and so are safe even if copying throws, as are the adaptors that use those operations.

The iterators returned from standard containers need to be copyable without throwing so that it is possible to roll back insertions by calling *erase()*.

Working Paper Changes

Replace the last paragraph of section 23.1 (Container requirements) with the following paragraph:

> Unless otherwise specified (see 23.2.1.3 and 23.2.4.3) all container types defined in this clause meet the following additional requirements:
>
> - if an exception is thrown by an **`insert()`** function while inserting a single element that function has no effects.
> - if an exception is thrown by a **`push_back()`** or **`push_front()`** function that function has no effects.
> - no **`erase()`**, **`pop_back()`** or **`pop_front()`** function throws an exception.
> - no copy constructor or assignment operator of a returned iterator throws an exception.
> - no **`swap()`** function throws an exception unless that exception is thrown by the copy constructor or assignment operator of the container's **`Compare`** object (if any, see 23.1.2).
> - no **`swap()`** function invalidates any references, pointers, or iterators referring to the elements of the sequences being swapped.

Insert the following paragraph after paragraph 1 of section 23.2.1.3 (*deque::insert*):

> Notes: If an exception is thrown other than by the copy constructor or assignment operator of **`T`** there are no effects.

Insert the following paragraph after paragraph 3 of section 23.2.1.3 (*deque::erase*):

> Throws: Nothing unless an exception is thrown by the copy constructor or assignment operator of **`T`**.

Append the following sentence to paragraph 1 of section 23.2.4.3 (*vector::insert*):

> If an exception is thrown other than by the copy constructor or assignment operator of **`T`** there are no effects.

Insert the following paragraph after paragraph 3 of section 23.2.4.3 (*vector::erase*):

> Throws: Nothing unless an exception is thrown by the copy constructor or assignment operator of **`T`**.