Removing Globals construct() and destroy() (Revision 1)

by Nathan Myers <myersn@roguewave.com>
Rogue Wave Software

08-Mar-95
X3J16/95-0039R1
WG21/N0639R1

In Clause 20, a number of apparently unnecessary global functions are declared, including:

```
  template <class T>
    inline T* allocate(ptrdiff_t, T*);

  template <class T>
    inline void deallocate(T* buffer);

  template <class T, class T2>
    inline void construct(T* p, const T2& value);

  template <class T>
    inline void destroy(T* pointer);

  template <class T>
    pair<T*, ptrdiff_t> get_temporary_buffer(ptrdiff_t, T*);
```

While these functions may once have been useful in implementing STL without allocator, they are not necessary for the purposes of the Standard Library, in that they do not aid communication between modules, implement semantics that cannnot reproduced by users, or provide substantial functionality. Furthermore, they are subsumed by STL allocator members, which are what STL-conforming collections are expected to use instead.

(Amendment 1)

Some have suggested that, because the purpose of these functions was in support of writers of STL-like collections, removing them outright is not so obviously correct. Furthermore, implementors of object databases and garbage collecting stores have remarked that providing these functions as allocator operations would allow them to be (partially) specialized them to take greater control of construction and destruction semantics.

Therefore, I have accepted as a friendly amendment, rather than removing all the above functions, to integrate them with allocator. They are described in 95-0019R1/N0619R1.

(I have not generalized destroy<FwdIter>(FwdIter first, FwdIter last) for use with allocator because this operation seems incompatible with allocator models of memory management, and because the template deductions required are not supported by the language.)