Document No: N3106 = 10-0096
Date: 2010-08-05
Project: Programming Language C++
Reply to: Nicolai Josuttis
nico@josuttis.de

Proposed Resolution for US 122: Revision of N2772 and Issue 915 to adopt it into the Standard

Rationale

In Pittsburgh we accepted the formal motion to N2772, but it didn't make it into the Draft Standard, due to unclear meaning for the old wording in respect to the current version of the Standard. This paper provides concrete wording for all places of the modification.

Proposed Wording

In **25.1 General,** Header <algorithm> synopsis

Replace

```
template<class T> const T& min(const T& a, const T& b);

template<class T> const T& min(const T& a, const T& b, const T& c);

template<class T, class... Args>
const T& min(const T& a, const Args&... args);

template<class T, class Compare>
const T& min(const T& a, const T& b, Compare comp);

template<class T, class U, class... Args>
const T& min(const T& a, const U& b, const Args&... args);
```

by

```
template<class T> const T& min(const T& a, const T& b);

template<class T, class Compare>
const T& min(const T& a, const T& b, Compare comp);

template<class T>
T min(initializer_list<T> t);

template<class T, class Compare>
T min(initializer_list<T> t, Compare comp);
```

Replace

```
template<class T> const T& max(const T& a, const T& b);

template<class T> const T& max(const T& a, const T& b, const T& c);

template<class T, class... Args>
const T& max(const T& a, const Args&... args);

template<class T, class Compare>
const T& max(const T& a, const T& b, Compare comp);

template<class T, class U, class... Args>
const T& max(const T& a, const U& b, const Args&... args);
```

by

```
template<class T>
const T& max(const T& a, const T& b);

template<class T, class Compare>
const T& max(const T& a, const T& b, Compare comp);

template<class T>
T max(initializer_list<T> t);

template<class T, class Compare>
T max(initializer_list<T> t, Compare comp);
```

Replace

```
template<class T> pair<const T&, const T&> minmax(const T& a, const T& b);

template<class T> pair<const T&, const T&> minmax(const T& a, const T& b,
                                                  const T& c);

template<class T, class... Args>
pair<const T&, const T&> minmax(const T& a, const Args&... args);

template<class T, class Compare>
pair<const T&, const T&> minmax(const T& a, const T& b, Compare comp);

template<class T, class U, class... Args>
pair<const T&, const T&> minmax(const T& a, const U& b, const Args&... args);
```

by

```
template<class T>
pair<const T&, const T&> minmax(const T& a, const T& b);

template<class T, class Compare>
pair<const T&, const T&> minmax(const T& a, const T& b, Compare comp);

template<class T>
pair<T,T> minmax(initializer_list<T> t);

template<class T, class Compare>
pair<T,T> minmax(initializer_list<T> t, Compare comp);
```

In **25.4.7 Minimum and maximum**

Replace

```
{template<class T> const T&
min(const T& a, const T& b, const T& c);

template<class T, class... Args>
const T& min(const T& a, const Args&... args);
```

4 Requires: T is LessThanComparable, and all types forming Args... are the same as T.

5 Returns: The smallest value in the set of all the arguments.

6 Remarks: Returns the leftmost argument when several arguments are equivalent to the smallest. Returns a if sizeof...(Args) is 0.

```
template<class T, class U, class... Args>
const T& min(const T& a, const U& b, const Args&... args);
```

7 Requires: The types of all the arguments except the last one are the same as T. The last argument is a binary predicate over T.

8 Returns: The first element in a partial ordering of all the arguments except the last one, where the ordering is defined by the predicate.

9 Remarks: Returns the leftmost argument when several arguments are equivalent to the first element in the ordering. Returns a if sizeof...(Args) is 0.

By:

```
template<class T>
T min(initializer_list<T> t);

template<class T, class Compare>
T min(initializer_list<T> t, Compare comp);
```

Requires: T is LessThanComparable and CopyConstructible and t.size()>0.

Returns: The smallest value in the initializer_list.

Remarks: Returns a copy of the leftmost argument when several arguments are equivalent to the smallest.

Replace

```
template<class T> const T&
max(const T& a, const T& b, const T& c);

template<class T, class... Args>
const T& max(const T& a, const Args&... args);
```

13 Requires: T is LessThanComparable, and all types forming Args... are the same as T.

14 Returns: The largest value in the set of all the arguments.

15 Remarks: Returns the leftmost argument when several arguments are equivalent to the largest. Returns a if sizeof...(Args) is 0.

```
template<class T, class U, class... Args>
const T& max(const T& a, const U& b, const Args&... args);
```

16 Requires: The types of all the arguments except the last one are the same as T. The last argument is a binary predicate over T.

17 Returns: The last element in a partial ordering of all the arguments except the last one, where the ordering is defined by the predicate.

18 Remarks: Returns the leftmost argument when several arguments are equivalent to the first element in the ordering. Returns a if sizeof...(Args) is 0.

By:

```
template<class T>
T max(initializer_list<T> t);

template<class T, class Compare>
T max(initializer_list<T> t, Compare comp);
```

Requires: T is LessThanComparable and CopyConstructible and t.size()>0.

Returns: The largest value in the initializer_list.

Remarks: Returns a copy of the leftmost argument when several arguments are equivalent to the largest.

Replace

```
template<class T> pair<const T&, const T&>
minmax(const T& a, const T& b, const T& c);

template<class T, class... Args>
pair<const T&, const T&> minmax(const T& a, const Args&... args);
```

23 Requires: T is LessThanComparable, and all types forming Args... are the same as T.

24 Returns: pair<const T&, const T&>(x, y) where x is the first element and y is the last element in a partial ordering of all the arguments.

25 Remarks: x is the leftmost argument when several arguments are equivalent to the smallest. y is the rightmost argument when several arguments are equivalent to the largest. Returns pair<const T&, const T&>(a, a) if sizeof...(Args) is 0.

26 Complexity: At most (3/2) sizeof...(Args) applications of the corresponding predicate.

```
template<class T, class U, class... Args>
pair<const T&, const T&> minmax(const T& a, const U& b,
                                const Args&... args);
```

27 Requires: The types of all the arguments except the last one are the same as T. The last argument is a binary predicate over T.

28 Returns: pair<const T&, const T&>(x, y) where x is the first element and y is the last element in a partial ordering of all the arguments except the last one, where the ordering is defined by the predicate.

29 Remarks: x is the leftmost argument when several arguments would order equivalent as first in the ordering. y is the rightmost argument when several arguments would order equivalent as last in the ordering. Returns pair<const T&, const T&>(a, a) if sizeof...(Args) is 0.

30 Complexity: At most (3/2) sizeof...(Args) applications of the corresponding predicate.

By:

```
template<class T>
pair<T,T> minmax(initializer_list<T> t);

template<class T, class Compare>
pair<T,T> minmax(initializer_list<T> t, Compare comp);
```

Requires: T is LessThanComparable and CopyConstructible and t.size()>0.

Returns: pair<T,T>(x, y) where x has the smallest and y has the largest value in the initializer_list.

Remarks: x is a copy of the leftmost argument when several arguments are equivalent to the smallest. y is a copy of the rightmost argument when several arguments are equivalent to the largest.

Complexity: At most (3/2) * t.size() applications of the corresponding predicate.