Revision 2 of:
Proposed Resolution for CH 15:
Double check copy and move semantics of classes due to new rules for default move constructors and assignment operators

Rationale

In Pittsburgh the default semantics of move constructors and move assignment operators changed. This might have the effect that accidentally we enable or disable move and/or copy semantics unintentionally.

Note also that this proposed resolution adds missing constructor descriptions for stack<>.

In contrast to N3112 this paper:

- Skips changes already done in the Standard (e.g. pair)
- Skips changes that became obsolete in the meantime (e.g. atomic_future)
- Do nothing on basic_streambuf<>
- Skips changes where only a deleted copy constructor/assignment exists and no move semantics is specified (because this implies a deletes move constructor/assignment)
- Updates section numbers according to latest draft
In the library we found the following places where move semantics are explicitly specified and copy semantics differ and discussed them in the library group. The following table gives an overview of the result of the discussion, according to the status of the library in the Madrid meeting:

- **Red flags** indicate places where we have to fix something.
- **Yellow flags** indicate places where we don’t have to fix something. But to clarify we might add explicit statements (usually with =delete) indicating that we disable copying/moving by intention. The proposed solution in this paper contains proposed wording for these places.
- **Green flags** indicate places where we intentionally have different semantics and don’t have to do anything.

<table>
<thead>
<tr>
<th>Section</th>
<th>Sec.-No</th>
<th>Class</th>
<th>C(const&amp;)</th>
<th>C(&amp;&amp;)</th>
<th>op=(const&amp;)</th>
<th>op=(&amp;&amp;)</th>
<th>Proposed resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>[allocator.adaptor.syn]</td>
<td>20.12.1</td>
<td>scoped_allocator_adaptor&lt;&gt;</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>add move constr. with specified behavior</td>
</tr>
<tr>
<td>[queue.defn]</td>
<td>23.6.3.1</td>
<td>queue&lt;&gt;</td>
<td>yes</td>
<td></td>
<td>yes</td>
<td></td>
<td>remove move versions (all default)</td>
</tr>
<tr>
<td>[priority.queue]</td>
<td>23.6.4</td>
<td>priority_queue&lt;&gt;</td>
<td>yes</td>
<td></td>
<td>yes</td>
<td></td>
<td>remove move versions (all default)</td>
</tr>
<tr>
<td>[stack.defn]</td>
<td>23.6.5.2</td>
<td>stack&lt;&gt;</td>
<td>red</td>
<td>red</td>
<td>yes</td>
<td></td>
<td>remove move versions (all default)</td>
</tr>
<tr>
<td>[istream]</td>
<td>27.7.2.1</td>
<td>basic_istream&lt;&gt;</td>
<td>protected</td>
<td></td>
<td></td>
<td></td>
<td>add =delete for copy semantics</td>
</tr>
<tr>
<td>[iostreamclass]</td>
<td>27.7.2.5</td>
<td>basic_iostream&lt;&gt;</td>
<td>protected</td>
<td></td>
<td></td>
<td></td>
<td>add =delete for copy semantics</td>
</tr>
<tr>
<td>[ostream]</td>
<td>27.7.3.1</td>
<td>basic_ostream&lt;&gt;</td>
<td>protected</td>
<td></td>
<td></td>
<td></td>
<td>add =delete for copy semantics</td>
</tr>
<tr>
<td>[stringstream]</td>
<td>27.8.3</td>
<td>basic_ostreamstream&lt;&gt;</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>add =delete for copy semantics</td>
</tr>
<tr>
<td>[ostringstream]</td>
<td>27.8.4</td>
<td>basic_ostringstream&lt;&gt;</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>add =delete for copy semantics</td>
</tr>
<tr>
<td>[stringstream]</td>
<td>27.8.5</td>
<td>basic_stringstream&lt;&gt;</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>add =delete for copy semantics</td>
</tr>
<tr>
<td>[ifstream]</td>
<td>27.9.1.6</td>
<td>basic_ifstream&lt;&gt;</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>add =delete for copy semantics</td>
</tr>
<tr>
<td>[ofstream]</td>
<td>27.9.1.10</td>
<td>basic_ofstream&lt;&gt;</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>add =delete for copy semantics</td>
</tr>
<tr>
<td>[fstream]</td>
<td>27.9.1.14</td>
<td>basic_fstream&lt;&gt;</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>add =delete for copy semantics</td>
</tr>
<tr>
<td>[streambuf]</td>
<td>27.6.3</td>
<td>basic_streambuf&lt;&gt;</td>
<td>protected</td>
<td></td>
<td></td>
<td></td>
<td>do nothing (we intend to have no move operations, which is different than to delete them)</td>
</tr>
<tr>
<td>[stringbuf]</td>
<td>27.8.2</td>
<td>basic_stringbuf&lt;&gt;</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>add =delete for copy semantics</td>
</tr>
<tr>
<td>[filebuf]</td>
<td>27.9.1.1</td>
<td>basic_filebuf&lt;&gt;</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td>add =delete for copy semantics</td>
</tr>
</tbody>
</table>
Proposed Wording for Scoped Allocator Adaptor

In **20.12.1 Header** <scoped_allocator> *synopsis [allocator.adaptor.syn]* in the declaration of class scoped_allocator_adaptor after
   
   scoped_allocator_adaptor(const scoped_allocator_adaptor& other);

**add**
   
   scoped_allocator_adaptor(scoped_allocator_adaptor&& other);

In **20.12.3 Scoped allocator adaptor constructors [allocator.adaptor.cnstr]** after § 4 (copy constructor)

**add**:

   scoped_allocator_adaptor(scoped_allocator_adaptor&& other);

*Effects*: Move constructs each allocator within the adaptor with the corresponding allocator from other.

Editorial comment:

   In **20.12.3 Scoped allocator adaptor constructors [allocator.adaptor.cnstr]** §4 replace “initializes” by “initializes”.

Proposed Wording for Container Adaptors

In **23.6.3.1 queue definition [queue.defn]**

**strike**

   queue(queue&& q);

and **strike**

   queue& operator=(queue&& q);

In **23.6.3.2 queue constructors [queue.cons]**

**strike**:

   queue(queue&& q);

   3 *Effects*: Initializes c with std::move(q,c).

   queue& operator=(queue&& q);

   4 *Effects*: Assigns std::move(q,c) to c.

   5 Returns: "this."
Editorial comment:

In 23.6.3.2 queue constructors [queue.cons] §1 replace “Initializes” by “Initializes”.

In 23.6.4 Class template priority_queue [priority.queue]
strike
priority_queue(priority_queue&&);
and strike
priority_queue& operator=(priority_queue&&);

In 23.6.4.1 priority_queue constructors [priqueue.cons]
strike:

priority_queue(priority_queue&& q);

5 Effects: Initializes c with std::move(q.c) and initializes comp with std::move(q.comp).

priority_queue& operator=(priority_queue&& q);

6 Effects: Assigns std::move(q.c) to c and assigns std::move(q.comp) to comp.

7 Returns: *this.

In 23.6.5.2 stack definition [stack.defn]
strike
stack(stack&&s);
and strike
stack& operator=(stack&& s);

In 23.6.5.3 stack constructors [stack.cons]
strike:

stack(stack&& s);
Effects: Initializes c with std::move(s.c).

stack& operator=(stack&& s);

1 Effects: Assigns std::move(s.c) to c.

2 Returns: *this.

And add:

explicit stack(const Container& cont);
1 Effects: Initializes c with cont.

explicit stack(Container&& cont = Container());

2 Effects: Initializes c with std::move(cont).
Proposed Wording for IO-Streams

In **27.7.1 Class template basic_istream [istream]**
before

```
basic_istream(basic_istream&& rhs);
```

add:

```
basic_istream(const basic_istream& rhs) = delete;
```

and before:

```
basic_istream& operator=(basic_istream&& rhs);
```

add:

```
basic_istream& operator=(const basic_istream& rhs) = delete;
```

In **27.7.2.5 Class template basic_iostream [iostreamclass]**
before

```
basic_iostream(basic_iostream&& rhs);
```

add:

```
basic_iostream(const basic_iostream& rhs) = delete;
```

and before:

```
basic_iostream& operator=(basic_iostream&& rhs);
```

add:

```
basic_iostream& operator=(const basic_iostream& rhs) = delete;
```

In **27.7.3.1 Class template basic_ostream [ostream]**
before

```
basic_ostream(basic_ostream&& rhs);
```

add:

```
basic_ostream(const basic_ostream& rhs) = delete;
```

and before:

```
basic_ostream& operator=(basic_ostream&& rhs);
```

add:

```
basic_ostream& operator=(const basic_ostream& rhs) = delete;
```

In **27.7.8.3 Class template basic_istringstream [istringstream]**
before

```
basic_istringstream(basic_istringstream&& rhs);
```

add:

```
basic_istringstream(const basic_istringstream& rhs) = delete;
```

and before:

```
basic_istringstream& operator=(basic_istringstream&& rhs);
```

add:

```
basic_istringstream& operator=(const basic_istringstream& rhs) = delete;
```
In 27.8.4 Class template basic_ostringstream [ostringstream]
before
basic_ostringstream(basic_ostringstream&& rhs);
add:
basic_ostringstream(const basic_ostringstream& rhs) = delete;
and before:
basic_ostringstream& operator=(basic_ostringstream&& rhs);
add:
basic_ostringstream& operator=(const basic_ostringstream& rhs) = delete;

In 27.8.5 Class template basic_stringstream [stringstream]
before
basic_stringstream(basic_stringstream&& rhs);
add:
basic_stringstream(const basic_stringstream& rhs) = delete;
and before:
basic_stringstream& operator=(basic_stringstream&& rhs);
add:
basic_stringstream& operator=(const basic_stringstream& rhs) = delete;

In 27.9.1.6 Class template basic_ifstream [ifstream]
before
basic_ifstream(basic_ifstream&& rhs);
add:
basic_ifstream(const basic_ifstream& rhs) = delete;
and before:
basic_ifstream& operator=(basic_ifstream&& rhs);
add:
basic_ifstream& operator=(const basic_ifstream& rhs) = delete;

In 27.9.1.10 Class template basic_ofstream
before
basic_ofstream(basic_ofstream&& rhs);
add:
basic_ofstream(const basic_ofstream& rhs) = delete;
and before:
basic_ofstream& operator=(basic_ofstream&& rhs);
add:
basic_ofstream& operator=(const basic_ofstream& rhs) = delete;

In 27.9.1.14 Class template basic_fstream
before
basic_fstream(basic_fstream&& rhs);
add:
basic_fstream(const basic_fstream& rhs) = delete;
and before:
basic_fstream& operator=(basic_fstream&& rhs);
add:
basic_fstream& operator=(const basic_fstream& rhs) = delete;
In 27.8.2 Class template basic_stringbuf [stringbuf]
before
   basic_stringbuf(basic_stringbuf&& rhs);
add:
   basic_stringbuf(const basic_stringbuf& rhs) = delete;
and before:
   basic_stringbuf& operator=(basic_stringbuf&& rhs);
add:
   basic_stringbuf& operator=(const basic_stringbuf& rhs) = delete;

In 27.9.1.1 Class template basic_filebuf [filebuf]
before
   basic_filebuf(basic_filebuf&& rhs);
add:
   basic_filebuf(const basic_filebuf& rhs) = delete;
and before:
   basic_filebuf& operator=(basic_filebuf&& rhs);
add:
   basic_filebuf& operator=(const basic_filebuf& rhs) = delete;