```
struct S1 { S1(S1&&); };             // move constructor
struct S2 { S2(const S2&&); };       //   "       "
struct S3 { S3(S3&&, int i = 0); };  //   "       "
struct S4 { S4(S4&&, int i); };      // not a move constructor
struct S5 { S5(int&&); };            //  "  "   "       "
struct S6 { S6(S6&); };              //  "  "   "       "
```

2. A move-assignment operator for a type `X` is a nonstatic, nontemplate member function named **`operator=`** with exactly one parameter that is a cv-qualified rvalue reference to `X`, i.e., either `X&&`, **`const`** `X&&`, **`volatile`** `X&&`, or **`const volatile`** `X&&`. Any return type and value is valid for a move-assignment operator, but the common convention is to have a return type of `X&` and to return **`*this`**.

Just as with other special member functions, when not declared explicitly, it is possible for the move constructor and move-assignment operator to be declared implicitly[10] for a **`class`**, or **`struct`**, `X`.

- Supplying any of a **user-declared** copy or move constructor, copy or move assignment operator, or destructor will suppress the implicit generation of both move operations.
- The default move constructor will have the **function prototype** `X::X(X&&)`. The default move-assignment operator will have the signature `X&` **`operator=`**`(X&&)`.
- The **exception specification** and **triviality** of both move operations are determined by the exception specification and triviality of the corresponding operation on all base classes and non**`static`** data members of `X`.
- The default implementation will apply the corresponding operation to each base class and non**`static`** data member.

The rules governing special-member-function generation for a **`union`** are similar to those for a **`class`** or **`struct`** with the added proviso that *any* of the **`union`**'s six special member functions that are not declared explicitly will be **deleted** (see Section 1.1."Deleted Functions" on page 53) if they correspond to a *non***trivial** special member function of one or more of the **`union`**'s *non***`static`** data members:

```
struct S { S(S&&); };  // S has a user-provided (nontrivial) move ctor.
union U { S s; };      // U's implicitly declared move ctor is deleted.
```

Note that a **trivial move operation** and a **trivial copy operation** on a union have the same behavior: a bitwise copy. For a much more detailed discussion of both **`union`**s and triviality, see Section 3.1."**`union`** '11" on page 1174 and Section 2.1."Generalized PODs '11" on page 401.

[10] For more on the implicit generation of special member functions, see Section 1.1."Defaulted Functions" on page 33.