Title: Open Issues for Numeric Libraries (Chapter 26)
Author: Judy Ward
Document Number: X3J16/95-0110
WG21/N0710

Work Group: Library
Issue Number: 26/001
Title: operator conversions in complex
Section: 26.1 New
Status: active
Description:

To: C++ libraries mailing list
Message c++std-lib-3382

```
/*

There seems to be a problem with the current interface
of the complex library. At Valley Forge
complex was changed to a templatized library. This
is an example of code that would have worked with
the non-templatized float_complex class but will no
longer work with the templatized complex<float> class.
Shouldn't this code still be valid?

Judy Ward
ward@roguewave.com
*/

class float_complex {}; // old non-templatized float complex class
float_complex operator/(float,float_complex);

template <class T>
class complex {}; // new templatized complex class
template <class T>
complex<T> operator/(const T&,const complex<T>&);

void main() {
      float_complex fcf;
      complex<float> cf;
      fcf=1/fcf; // ok
      cf=1/cf; // can't match
}
```

Resolution:

Requestor: Judy Ward

From John Spicer at EDG:

```
>The complex template should work okay if the operator functions are
>declared as friends inside the class definition.  These friend
>functions participate in overloading as normal functions do.
>Consequently, the normal conversions can be performed on their
>arguments.
>
>The example below should do what you want it to.
>
>John.
>
>template <class T> struct complex {
>     friend complex operator/(const T&, const complex);
>};
```

```
>
>int main()
>{
>     complex<float> cf;
>     cf = 1 / cf;
>}
```

If we use this approach, we would have to specify that all the operator declarations listed in Section 26.2 must be friends.

Owner: Judy Ward
Emails: (email reflector messages that discuss this issue)
c++std-lib-3385
c++std-ext-2832
c++std-lib-3386
c++std-lib-3382
c++std-lib-3387
c++std-ext-2833 (same as above)
c++std-lib-3771
Papers: (committee documents that discuss this issue)

Work Group: Library
Issue Number: 26/002
Title: complex needs to be updated for new iostreams
Section: 26 New
Status: active
Description:
the complex library's insertion/extraction operators need to

be updated for the new iostreams, i.e.

from:

```
istream& operator>>(istream&,complex<T>&);
ostream& operator<<(ostream&,const complex<T>&);
```

to:

```
template<class T, class charT, class traits>
basic_istream<charT, traits>& operator>>(basic_istream<charT, traits>&,
complex<T>&);

template<class T, class charT, class traits>
basic_ostream<charT, traits>& operator<<(basic_ostream<charT, traits>&, const
complex<T>&);
```

Resolution:
Requestor: Judy Ward
Owner: Judy Ward
Emails: (email reflector messages that discuss this issue)
Papers: (committee documents that discuss this issue)

Work Group: Library
Issue Number: 26/003
Title: complex library operator << needs to be refined
Section: 26 New
Status: active
Description:

To: C++ libraries mailing list
Message c++std-lib-3659

26.2.1.3.8 says operator<<(ostream &os, complex x) returns
os << '(' << x.real() << ',' << x.imag() << ')'

I take it the output from:

```
      complex x(4.5,2.2) ;
      cout << ':' << setiosflags(ios::left)
            << setw(12) << x ;
```

      would be
            :(           4.5,2.2)

It seems to me that complexes will behave more like the builtin types
if operator<< did something like this:

```
ostream & operator<<(ostream &o, complex x) {
      ostrstream ost; // should be stringstream
      ost.precision(o.precision()); // and other flags too
      ost << '(' << x.real() << ',' << x.imag() << ')' << ends ;
      o << ost.str() ;
      ost.rdbuf()->freeze(0) ;
      return o;
}
```

If this has been discussed and rejected as too busy by the library
group, please forgive my intrusion.

 Resolution:
 Requestor:    Thomas Holday, JP Morgan
 Owner:         Judy Ward
 Emails: (email reflector messages that discuss this issue)
c++std-lib-3662
c++std-lib-3665
c++std-lib-3676
c++std-lib-3678
c++std-lib-3681
c++std-lib-3682
 Papers: (committee documents that discuss this issue)

 Work Group:     Library
 Issue Number:   26/004
 Title: cleanup of Chapter 26
 Section:        26 New
 Status:         active
 Description:

Hopefully these are all editorial changes. I noticed
the following typos and mistakes.

Section 26.2.5

1. There is an extra "P" after lhs in ther Returns for operator==

2. The "Returns" for operator!= should be:

rhs.real() != lhs.real() || rhs.imag() != lhs.imag()

Section 26.2.6

1. In the "Returns" for arg and conj, what is TBS (To be specified?)?
   I would make them:

   for arg, Returns the phase angle of x.
   for conj, Returns the conjugate of x.

2. The polar function

   change t to T in the second arg
   The second argument used to default to 0 -- has that changed?

Section 26.2.7
   In the description, I don't think the "F" means anything, I'd remove it.

Section 26.3
    the "an" should be "a"

Section 26.5
     The added signatures list at the end is incomplete.
     It only includes the the float ones, not the long
     double functions mentioned in the previous sentence.

     Why does the double abs(double) have a comment that says fabs?
     I'm not sure what this comment or the labs() or ldiv() comments
     mean.

     Shouldn't the last two prototypes be:
     float abs(float);
     float pow(float, int);

Section 26.3.1.2
      the argument to operator= should be:
      const valarray<T>&  (not: const valarra<T>y&)

 Resolution:
 Requestor:  Judy Ward
 Owner:         Judy Ward
 Emails: (email reflector messages that discuss this issue)
 Papers: (committee documents that discuss this issue)

 Work Group:     Library
 Issue Number:   26/005
 Title: exceptions in valarray classes
 Section:        26 New
 Status:         active
 Description:

Box 125 says the description of valarray and its classes
lack any discussion of possible exceptions. There is a paragraph
above this issue which says bad_allocs are allowed. Is there
anything else that people want to add or should we close this
issue?

 Resolution:
 Requestor:  Judy Ward
 Owner:         Judy Ward
 Emails: (email reflector messages that discuss this issue)

**Papers: (committee documents that discuss this issue)**