

## Re: Difficulty with nested template classes (newbie)

*Source:* [http://coding.derkeiler.com/Archive/C\\_CPP/comp.lang.cpp/2003-11/3289.html](http://coding.derkeiler.com/Archive/C_CPP/comp.lang.cpp/2003-11/3289.html)

---

*From:* Zenon ([zenonk\\_at\\_comcast.net](mailto:zenonk_at_comcast.net))

*Date:* 11/26/03

Date: 26 Nov 2003 14:31:28 -0800

Thanks for the help. Using your suggestions and quite a few hours of playing around, I have gotten my program to the point where none of the obvious errors are being flagged in the template class, but instead in the driver class (which I know has no errors). Sorry for the huge post but I am including first the errors, then the template class and finally the driver. Thakns again for any help folks.

Zenon

Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland  
ex5.cpp:  
Error E2285 ex5.cpp 141: Could not find a match for  
'matrix<complex<double> >::o  
perator =(complex<double>)' in function main()  
Error E2285 ex5.cpp 142: Could not find a match for  
'matrix<complex<double> >::o  
perator =(complex<double>)' in function main()  
Error E2285 ex5.cpp 143: Could not find a match for  
'matrix<complex<double> >::o  
perator =(complex<double>)' in function main()  
Error E2285 ex5.cpp 144: Could not find a match for  
'matrix<complex<double> >::o  
perator =(complex<double>)' in function main()  
Error E2285 ex5.cpp 176: Could not find a match for  
'complex<double>::operator =  
<X>(matrix<complex<double> >)' in function main()  
Error E2034 ex5.cpp 185: Cannot convert 'matrix<double>' to 'double'  
in function  
main()  
Error E2034 ex5.cpp 242: Cannot convert 'const matrix<double>' to  
'matrix<Type>'  
in function main()  
Error E2342 ex5.cpp 242: Type mismatch in parameter 'object' (wanted  
'const matr  
ix<Type> &', got 'matrix<double>') in function main()  
Error E2102 ex5.cpp 248: Cannot use template 'yp' without specifying

```
specializat
ion parameters in function main()
Error E2102 ex5.cpp 250: Cannot use template 'yp' without specifying
specializat
ion parameters in function main()
Error E2102 ex5.cpp 251: Cannot use template 'yp' without specifying
specializat
ion parameters in function main()
Error E2357 matrix.h 318: Reference initialized with 'double', needs
lvalue of t
ype 'matrix<double>' in function matrix<double>::lookup(int,int)
Warning W8004 matrix.h 319: 'linear_index' is assigned a value that is
never use
d in function matrix<double>::lookup(int,int)
Error E2357 matrix.h 326: Reference initialized with 'const double',
needs lvalu
e of type 'matrix<double>' in function matrix<double>::lookup(int,int)
const
Warning W8004 matrix.h 327: 'linear_index' is assigned a value that is
never use
d in function matrix<double>::lookup(int,int) const
Error E2357 matrix.h 326: Reference initialized with 'const
complex<double>', ne
eds lvalue of type 'matrix<complex<double> >' in function
matrix<complex<double>
>::lookup(int,int) const
Warning W8004 matrix.h 327: 'linear_index' is assigned a value that is
never use
d in function matrix<complex<double> >::lookup(int,int) const
*** 14 errors in Compile ***
```

```
// Template class
#include <iostream>
#include <string>
#include <vector>

template <class Type>
class matrix
{
private:
    std::vector<Type> _data ;
public:

    class iterator
    {
        int _current_row ;
        int _current_col ;
        matrix<Type>& _object ;

    public:
        // Iterator constructor implementation
```

comp.lang.c++. Re: Difficulty with nested template classes (newbie)

```
iterator (matrix<Type>& object)
    : _object (object), _current_row (0), _current_col (0)
    {}
~iterator ()
    {}
iterator operator++ (int dummy)
    {
        _current_col++;
        if (_current_col >= _object.cols())
            {
                _current_row++;
                _current_col = 0;
            }
        return *this;
    }

    static matrix<Type> dummy;
matrix<Type>& operator* ()
    {
        if (*this)
            {
                return _object (_current_row, _current_col);
            }
        else
            {
                return dummy;
            }
    }
operator bool () const
    {
        if (_current_row >= _object.rows())
            {
                return false;
            }
        else
            {
                return true;
            }
    }

};
//*****
template <class Type>
class const_iterator
{
    int _current_row;
    int _current_col;
    const matrix<Type>& _object;

public:
    const_iterator (const matrix<Type>& object)
```

comp.lang.c++. Re: Difficulty with nested template classes (newbie)

```
:_object (object), _current_row (0), _current_col (0)
{}

~const_iterator ()
{}

const_iterator operator++ (int dummy)
{
    _current_col++;
    if (_current_col >= _object.cols())
    {
        _current_row++;
        _current_col = 0;
    }
    return *this;
}

static matrix<Type> dummy;
const matrix<Type>& operator* () const
{
    if (*this)
    {
        return _object (_current_row, _current_col);
    }
    else
    {
        return dummy;
    }
}

operator bool () const
{
    if (_current_row >= _object.rows())
    {
        return false;
    }
    else
    {
        return true;
    }
}

};
//*****

private:
    int _nrows;
    int _ncols;

public:
```

```
matrix (int rows=1, int cols=1) ;
matrix (const matrix<Type>& rhs) ;
~matrix () {} ;

matrix& operator=(const matrix<Type>& rhs) ;

matrix& lookup (int row, int col) ;
const matrix& lookup (int row, int col) const ;

matrix& operator() (int row, int col)
{
    return lookup (row, col) ;
}
const matrix& operator() (int row, int col) const
{
    return lookup (row, col) ;
}

int rows () const
{
    return _nrows ;
}
int cols () const
{
    return _ncols ;
}

bool add (const matrix<Type>& rhs, matrix<Type>& result) const ;
matrix operator+ (const matrix<Type>& rhs) const ;

void put (std::ostream& os) const ;
void get (std::istream& is) ;

friend class iterator ;

} ; // end matrix

//*****

// Matrix constructor implementation
template <class Type>
matrix<Type>::matrix (int rows, int cols)
    : _data (rows * cols), _nrows (rows), _ncols (cols)
{ }

//
template <class Type>
matrix<Type>::matrix (const matrix<Type>& rhs)
    : _data (rhs._data), _nrows(rhs._nrows), _ncols(rhs._ncols)
{ }
```

```
// Matrix operator= implementation
template <class Type>
matrix<Type>& matrix<Type>::operator= (const matrix<Type>& rhs)
{
    if (this != &rhs)
    {
        _data = rhs._data ;
        _nrows = rhs._nrows ;
        _ncols = rhs._ncols ;
    }
    return *this ;
}

// Matrix lookup implementation
template <class Type>
matrix<Type>& matrix<Type>::lookup (int row, int col)
{
    int linear_index = row * _ncols + col ;
    return _data [linear_index] ;
}

// Overloaded matrix lookup implementation
template <class Type>
const matrix<Type>& matrix<Type>::lookup (int row, int col) const
{
    int linear_index = row * _ncols + col ;
    return _data [linear_index] ;
}

// Matrix Put implementation
template <class Type>
void matrix<Type>::put (std::ostream& os) const
{
    os << _nrows << " " << _ncols << endl ;

    for (int rix=0; rix<_nrows; rix++)
    {
        for (int cix=0; cix<_ncols; cix++)
        {
            os << lookup (rix, cix) << " " ;
        }
        os << endl ;
    }
}

// Matrix get implementation
template <class Type>
void matrix<Type>::get (std::istream& is)
{
    is >> _nrows >> _ncols ;
    _data.resize (_nrows * _ncols) ;
}
```

comp.lang.c++. Re: Difficulty with nested template classes (newbie)

```
for (int rix=0; rix<_nrows; rix++)
{
    for (int cix=0; cix<_ncols; cix++)
    {
        is >> lookup (rix, cix) ;
    }
}

// Matrix add implementation
template <class Type>
bool matrix<Type>::add (const matrix<Type>& rhs, matrix<Type>&
result) const
{
    if ((_nrows != rhs._nrows) || (_ncols != rhs._ncols))
    {
        return false ;
    }

    result._data.resize (_nrows * _ncols) ;

    for (int rix=0; rix<_nrows; rix++)
    {
        for (int cix=0; cix<_ncols; cix++)
        {
            result.lookup (rix, cix) = lookup (rix, cix) + rhs.lookup (rix,
cix) ;
        }
    }
    return true ;
}

// Matrix operator + implementation
template <class Type>
matrix<Type> matrix<Type>::operator+ (const matrix<Type>& rhs) const
{
    matrix<Type> result (_nrows, _ncols) ;
    add (rhs, result) ;
    return result ;
}

//*****

inline
template <class Type>
std::ostream& operator<< (std::ostream& os, const matrix<Type>& rhs)
{
    rhs. put (os) ;
    return os ;
}
```

```

inline
template <class Type>
std::istream& operator>> (std::istream& is, matrix<Type>& rhs)
{
    rhs.get (is) ;
    return is ;
}

// Driver
// =====
//
// File: ex5.C
//
// Description:
//
// This program is the "matrix" test driver for exercise 5. It
// creates
// several "matrix<double>" objects (some constant), copies them,
// and tries
// the new operators (and verifies the old functions too). It also
// creates
// a "matrix<complex<double>>" object and tries basic
// functionality.
//
// Verification is manual – that is, you run the program and
// inspect the output.
//
//
// =====
//

#include <fstream>
#include <iostream>
#include <complex>

#include "matrix.h"

using namespace std ;

//
// supporting function – this will be used in part 2 of the test
//
template <class T>
void print_matrix (ostream& os, const matrix<T>& data)
{
    int nrows = data.rows() ;
    int ncols = data.cols() ;

    os << nrows << " rows by " << ncols << " columns" << endl ;
}

```

```

for (int rix=0; rix<nrows; rix++)
{
    for (int cix=0; cix<ncols; cix++)
    {
        if (cix > 0)
        {
            os << ", " ;
        }
        os << data (rix, cix) ;
    }
    os << endl ;
}
}

int main ()
{

    ostream& os = cout ;

// -----
//
// Part 1 – test basic capabilities:
//
// constructor
// destructor
// operator ()
// operator <<
// iterator constructor
// iterator operator++
// iterator operator*
// iterator operator bool
//
// Create a matrix, fill it with data, and print it out.
//
// This test executes inside a block so that the "matrix"
// destructor is called at the end of part 1 testing.
//
// -----

{
    os << "Matrix Testing – Part 1" << endl ;
    os << "-----" << endl ;

    matrix<double> X (2, 3) ;

// operator ()

    X (0, 0) = 100.0 ;
    X (0, 1) = 100.1 ;
    X (0, 2) = 100.2 ;
    X (1, 0) = 101.0 ;

```

comp.lang.c++. Re: Difficulty with nested template classes (newbie)

```
X (1, 1) = 101.1 ;
X (1, 2) = 101.2 ;

// operator << and size

    int nrows = X.rows() ;
    int ncols = X.cols() ;
os << "matrix X (" << nrows << " rows by " << ncols
  << " colmns) = " << endl ;
os << X ;

//
// repeat the output operation using iterator
// (and change one matrix entry to verify that that works)
//

X (1, 1) = 201.1 ;
matrix<double>::iterator xp (X) ;
double value ;

os << endl ;
os << "Using iterator" << endl ;

while (xp) // operator bool
{
  os << *xp << endl ;
  xp++ ;
}

}

// -----
//
// Part 2 – Multiple objects:
//
// Now that we have confidence in the basic functionality of our
// matrix, let's create two of them and make sure there's no problem
// with that.
//
// -----

{
  os << endl ;
  os << endl ;
  os << "Matrix Testing – Part 2" << endl ;
  os << "-----" << endl ;

  matrix<complex<double> > Y (2, 2) ;

  Y (0, 0) = complex<double> (300.0, 1) ;
  Y (0, 1) = complex<double> (300.1, 0) ;
```

```
Y (1, 0) = complex<double> (301.0, -1) ;
```

```
Y (1, 1) = complex<double> (301.1, 0) ;
```

```
matrix<double> Z (3, 3) ;
```

```
Z (0, 0) = 400.0 ;
```

```
Z (0, 1) = 400.1 ;
```

```
Z (0, 2) = 400.2 ;
```

```
Z (1, 0) = 401.0 ;
```

```
Z (1, 1) = 401.1 ;
```

```
Z (1, 2) = 401.2 ;
```

```
Z (2, 0) = 402.0 ;
```

```
Z (2, 1) = 402.1 ;
```

```
Z (2, 2) = 402.2 ;
```

```
os << "Y = " << endl ;
```

```
print_matrix (os, Y) ;
```

```
os << endl ;
```

```
os << "Z = " << endl ;
```

```
print_matrix (os, Z) ;
```

```
//
```

```
// multiple iterators
```

```
//
```

```
matrix<complex<double> >::iterator yp (Y) ;
```

```
complex<double> yvalue ;
```

```
for (int yix=0; yix<2; yix++)
```

```
{
```

```
    yp++ ;
```

```
}
```

```
yvalue = *yp ;
```

```
matrix<double>::iterator zp (Z) ;
```

```
double zvalue ;
```

```
for (int zix=0; zix<6; zix++)
```

```
{
```

```
    zp++ ;
```

```
}
```

```
zvalue = *zp ;
```

```
os << endl ;
```

```
os << "*yp = " << yvalue << endl ;
```

```
os << "*zp = " << zvalue << endl ;
```

```
}
```

## comp.lang.c++. Re: Difficulty with nested template classes (newbie)

```
// -----  
//  
// Part 3 – Check copy operations and more of the const.  
//  
// -----  
  
{  
    os << endl ;  
    os << endl ;  
    os << "Matrix Testing – Part 3" << endl ;  
    os << "-----" << endl ;  
//  
// make a copy of a matrix using assignment, and using a copy  
// constructor.  
//  
    matrix<double> X (3, 2) ;  
  
// operator ()  
  
    X (0, 0) = 500.0 ;  
    X (0, 1) = 500.1 ;  
    X (1, 0) = 501.0 ;  
    X (1, 1) = 501.1 ;  
    X (2, 0) = 502.0 ;  
    X (2, 1) = 502.1 ;  
  
    const matrix<double> Y = X ; // copy constructor  
  
    matrix<double> Z ; // default constructor  
    Z = X ; // assignment  
  
    X (0, 0) = 100 ; // "destroy" X to ensure that  
    X (0, 1) = 100 ; // Y and Z are true copies  
    X (1, 0) = 100 ;  
    X (1, 1) = 100 ;  
    X (2, 0) = 100 ;  
    X (2, 1) = 100 ;  
  
    Z (0, 0) = 700.0 ;  
  
    os << "Y = " << endl ;  
    os << Y ;  
    os << endl ;  
    os << endl ;  
  
    //  
    // repeat the output operation using const_iterator  
    //  
  
    matrix<double>::const_iterator yp (Y) ;  
    double value ;
```

```
os << endl ;
os << "Using const_iterator" << endl ;

while (yp) // operator bool
{
    os << *yp << endl ;
    yp++ ;
}

os << "Z = " << endl ;
os << Z ;
os << endl ;
os << endl ;

matrix<double> S (Y+Z) ;
    matrix<double> T (Z+Y) ;

os << "S = " << endl ;
    os << S ;
    os << endl ;
    os << endl ;

os << "T = " << endl ;
    os << T ;
    os << endl ;
    os << endl ;

// check basic I/O

matrix<double> R ;
{
    ofstream otmp ("temp.tmp") ;
        otmp << S ;
        otmp << endl ;
    }

{
    ifstream itmp ("temp.tmp") ;
        itmp >> R ;
    }
os << "R = " << endl ;
    os << R ;
    os << endl ;
    os << endl ;

}

} // end exercise 3 test driver
```

comp.lang.c++. Re: Difficulty with nested template classes (newbie)

"Cy Edmunds" <cedmunds@spamless.rochester.rr.com> wrote in message  
news:<66Vvb.91004\$1N3.40300@twister.nyroc.rr.com>...  
> "Zenon" <zenonk@comcast.net> wrote in message  
> news:b1483a18.0311221755.6a808003@posting.google.com...  
>> I am having difficulty with nesting templates and the proper syntax.  
>> Can anyone please help? I'm sure it's obvious that I'm new to C++.  
>> Thanks in advance.  
>> -Zenon  
>>  
>> *template <class Type>*  
>> *class matrix*  
>> {  
>> *public:*  
>>  
>> *template <class Type>*  
>  
> *This line is a mistake -- the template Type is already in force here. Leave*  
> *this line out.*  
>  
>> *class iterator*  
>> {  
>> *int \_current\_row ;*  
>> *int \_current\_col ;*  
>> *matrix& \_object ;*  
>  
> *Should be:*  
>  
> *matrix<Type> & \_object;*  
>  
> *You don't need to qualify "matrix" in the constructor, but you do everywhere*  
> *else.*  
>  
>>  
>> *public:*  
>> *iterator (matrix& object) ;*  
>  
> *Same here:*  
> *iterator(matrix<Type> &object);*  
>  
>> *~iterator ()*  
>> {  
>> *iterator operator++ (int dummy) ;*  
>> *double& operator\* () ;*  
>> *operator bool () const ;*  
>> } ;  
>>  
>> *// Implementation of iterator template*  
>>  
>> *//Constructor implementation*  
>> *template <class Type>*  
>> *matrix<Type>::iterator<Type>::iterator (matrix<Type> & object)*

comp.lang.c++. Re: Difficulty with nested template classes (newbie)

```
>> : _object (object), _current_row (0), _current_col (0)
>> {}
>>
>> // Destructor implementation
>> template <class Type>
>> matrix<Type>::iterator<Type>::~iterator ()
>> {
>> delete _object;
>> }
>>
>> // Overloaded operator ++ implementation
>> matrix<Type>::iterator matrix<Type>::iterator<Type>::operator++ (int
>> dummy)
>> {
>> _current_col++;
>> if (_current_col >= _object. cols())
>> {
>> _current_row++;
>> _current_col = 0;
>> }
> return *this ;
>> }
>>
>> // Overloaded operator * implementation
>> template <class Type>
>> double& matrix<Type>::iterator::operator* ()
>> {
>> static double dummy ;
>> if (*this)
>> {
>> return _object (_current_row, _current_col) ;
>> }
>> else
>> {
>> return dummy ;
>> }
>> }
>>
>> // Overloaded operator bool implementation
>> template <class Type>
>> matrix<Type>::iterator::operator bool () const
>> {
>> if (_current_row >= _object. rows())
>> {
>> return false ;
>> }
>> else
>> {
>> return true ;
>> }
>> }
```

comp.lang.c++. Re: Difficulty with nested template classes (newbie)

>

> *I was too lazy to check your whole program. But I think my changes will make*

> *it a little closer to right. Good luck.*