

## Class "Header" File: atom.h

```

1  /**
2   * An atom class
3   */
4
5  #pragma once
6
7  #include <iostream> // system headers first
8  #include <string>
9
10 #include "nucleus.h" // developer headers last
11
12 using namespace std;
13
14 /**
15  * atom interface
16  */
17 class atom {
18 public:
19     // a constant data member for all instances
20     static const int MIDDLE_ATOMIC_NUMBER = 59;
21     // default ctor makes a Hydrogen atom
22     atom( );
23     atom( int num_pro_elec, int num_neutrons );
24
25     // get for a nucleus (returns a const reference!)
26     const nucleus& get_nucleus( ) const;
27     // accessors for changing neutrons — chained to nucleus accessors
28     void set_neutrons( int newvalue );
29     // move a neutron from other atom to calling object
30     void transfer_neutrons( atom& otheratom, int num );
31     // accessors for electrons
32     void set_electrons( int newvalue );
33     int get_electrons( ) const;
34     // move a electron from the other atom to calling object
35     void transfer_electrons( atom& otheratom, int num );
36     // get charge on the atom
37     int get_charge( ) const;
38
39     // read value from the input stream —
40     // returns a reference to the istream argument provided
41     istream& input( istream& is );
42     // print myself to the output stream —
43     // returns a reference to the ostream argument provided
44     ostream& output( ostream& os ) const;
45 private:
46     // helper function, makes sure electrons >=0
47     bool alter_electrons( int n );
48
49     nucleus nuc;
50     int electrons;
51 };

```

## Class Implementation File: atom.cxx

```

1  /**
2   * An atom class
3   */
4
5  #include <iostream>
6
7  #include "atom.h"
8
9  using namespace std;
10
11 /**
12  * atom implementation
13  */
14
15 // default ctor creates a H atom
16 atom::atom( )
17     : nuc( 1, 0 ), electrons(1)
18 {
19     // empty
20 }
21
22 atom::atom( int num_pro_elec, int num_neutrons )
23     : nuc( num_pro_elec, num_neutrons )
24 {
25     // make sure it matches nuc's contents if
26     // input values are invalid
27     electrons = nuc.get_protons();

```

## Class Implementation File: atom.cxx continued...

```
29 const nucleus& atom::get_nucleus( ) const
30 {
31     return nuc;
32 }
33
34 void atom::set_neutrons( int newvalue )
35 {
36     // chain to logic in the nucleus class
37     nuc.set_neutrons( newvalue );
38 }
39
40 void atom::transfer_neutrons( atom& otheratom,
41                             int num )
42 {
43     // chain to logic in the nucleus class
44     nuc.transfer_neutrons( otheratom.nuc, num );
45 }
46
47 void atom::set_electrons( int newvalue )
48 {
49     alter_electrons( newvalue );
50 }
51
52 int atom::get_electrons( ) const
53 {
54     return electrons;
55 }
56
57 // helper function
58 bool atom::alter_electrons( int n )
59 {
60     if( n >= 0 ) {
61         electrons = n;
62         return true;
63     }
64     return false;
65 }
66
67 void atom::transfer_electrons( atom& otheratom, int num )
68 {
69     if( num < 0 ) {
70         cout << "Negative_electron_transfer!" << endl;
71     } else if( otheratom.alter_electrons(otheratom.electrons-num)) {
72         // if they exist, move them
73         alter_electrons( electrons + num );
74     } else {
75         // ERROR!
76         cout << "Not_enough_electrons!" << endl;
77     }
78 }
79
80 int atom::get_charge( ) const
81 {
82     return nuc.get_protons() - electrons;
83 }
84
85 ostream& atom::output( ostream& os ) const
86 {
87     // Do not print an endl or '\n'!
88     nuc.output( os ) << " " << electrons;
89     return os;
90 }
91
92 }
```

## Class Implementation File: atom.cxx continued...

```
93
94 istream& atom::input( istream& is )
95 {
96     nucleus newnuc;
97     int newelectrons;
98
99     // format is NUCLEUS #electrons
100    if( newnuc.input( is ) >> newelectrons ) {
101        // if the input worked, AND the values are valid
102        if( newelectrons >= 0 ) {
103            nuc = newnuc;
104            electrons = newelectrons;
105        } else {
106            cout << "Invalid_number_of_electrons!" << endl;
107            // set the input stream to a failure state!
108            is.setstate( ios::failbit );
109        }
110    }
111    // returns is with failure state intact
112    return is;
113 }
```

## Driver "main": atom\_main.cxx

```
1  /**
2   * An example driver program for the simple atom class.
3   */
4  #include <iostream>
5
6  #include "atom.h"
7
8  using namespace std;
9
10 void mass_matters( const atom& a )
11 {
12     const nucleus& n( a.get_nucleus() ); // const ref
13     // show the use of a constant static data member
14     if( n.get_protons() < atom::MIDDLE_ATOMIC_NUMBER ) {
15         cout << n.symbol();
16         cout << "is probably lighter than half the";
17         cout << "known elements." << endl;
18     } else {
19         cout << a.get_nucleus().symbol();
20         cout << "has a big atomic number." << endl;
21     }
22 }
```

## Driver "main": atom\_main.cxx continued...

```

24 int main()
25 {
26     atom Carbon( 6, 6 );
27     atom Hydrogen;
28
29     cout << "This is carbon: ";
30     Carbon.output( cout ) << endl;
31
32     cout << "This is hydrogen: ";
33     Hydrogen.output( cout ) << endl;
34
35     // using get_nucleus - a function that returns
36     // a const reference
37     cout << "Hydrogen's symbol is ";
38     cout << Hydrogen.get_nucleus().symbol();
39     cout << endl;
40
41     cout << "Move a neutron from C to H..." << endl;
42     Hydrogen.transfer_neutrons( Carbon, 1 );
43     Hydrogen.output( cout ) << " ";
44     Carbon.output( cout ) << endl;
45
46     cout << "Move an electron from H to C..." << endl;
47     Carbon.transfer_electrons( Hydrogen, 1 );
48     Hydrogen.output( cout ) << " ";
49     Carbon.output( cout ) << endl;
50
51     cout << "Hydrogen charge is now: ";
52     cout << Hydrogen.get_charge() << endl;
53     cout << "Carbon charge is now: ";
54     cout << Carbon.get_charge() << endl;
55
56     // Relative descriptions of elements
57     mass_matters( Carbon );
58     mass_matters( atom( 82, 125 ) );
59
60     return 0;
61 }

```

## Example Driver Console Output (1 neutron transferred, electron transferred)

```

This is carbon: C 6 6 6
This is hydrogen: H 1 0 1
Hydrogen's symbol is H
Move a neutron from C to H...
H 1 0 1   C 6 5 6
Move an electron from H to C...
H 1 0 0   C 6 5 7
Hydrogen charge is now: 1
Carbon charge is now: -1
C is probably lighter than half the known elements.
Pb has a big atomic number.

```