Here's the output from STRCONV:

```
Joyeux Noel! Bonne Annee!
```

The STRCONV example demonstrates that conversions take place automatically not only in assignment statements but in other appropriate places, such as in arguments sent to operators (such as <<) or functions. If you supply an operator or a function with arguments of the wrong type, they will be converted to arguments of an acceptable type, provided you have defined such a conversion.

Note that you can't use an explicit assignment statement to convert a `String` to a C-string:

```
xstr = s2;
```

The C-string `xstr` is an array, and you can't normally assign to arrays (although as we'll see in Chapter 11, when you overload the assignment operator, all sorts of things are possible).

Conversions Between Objects of Different Classes

What about converting between objects of different user-defined classes? The same two methods just shown for conversions between basic types and user-defined types also apply to conversions between two user-defined types. That is, you can use a one-argument constructor or you can use a conversion operator. The choice depends on whether you want to put the conversion routine in the class declaration of the source object or of the destination object. For example, suppose you say

```
objecta = objectb;
```

where `objecta` is a member of class `A` and `objectb` is a member of class `B`. Is the conversion routine located in class `A` (the destination class, since `objecta` receives the value) or class `B` (the source class)? We'll look at both cases.

Two Kinds of Time

Our example programs will convert between two ways of measuring time: 12-hour time and 24-hour time. These methods of telling time are sometimes called *civilian time* and *military time*. Our `time12` class will represent civilian time, as used in digital clocks and airport flight departure displays. We'll assume that in this context there is no need for seconds, so `time12` uses only hours (from 1 to 12), minutes, and an "a.m." or "p.m." designation. Our `time24` class, which is for more exacting applications such as air navigation, uses hours (from 00 to 23), minutes, and seconds. Table 8.1 shows the differences.

TABLE 8.1 12-Hour and 24-Hour Time

12-Hour Time	*24-Hour Time*
12:00 a.m. (midnight)	00:00
12:01 a.m.	00:01
1:00 a.m.	01:00
6:00 a.m.	06:00
11:59 a.m	11:59
12:00 p.m. (noon)	12:00
12:01 p.m.	12:01
6:00 p.m.	18:00
11:59 p.m.	23:59

Note that 12 a.m. (midnight) in civilian time is 00 hours in military time. There is no 0 hour in civilian time.

Routine in Source Object

The first example program shows a conversion routine located in the source class. When the conversion routine is in the source class, it is commonly implemented as a conversion operator. Here's the listing for TIMES1:


```
//times1.cpp
//converts from time24 to time12 using operator in time24
#include <iostream>
#include <string>
using namespace std;
////////////////////////////////////////////////////////////////
class time12
   {
   private:
      bool pm;                        //true = pm, false = am
      int hrs;                        //1 to 12
      int mins;                       //0 to 59
   public:                            //no-arg constructor
      time12() : pm(true), hrs(0), mins(0)
         {  }
                                      //3-arg constructor
      time12(bool ap, int h, int m) : pm(ap), hrs(h), mins(m)
         {  }
      void display() const            //format: 11:59 p.m.
         {
         cout << hrs << ':';
```

```
         if(mins < 10)
            cout << '0';             //extra zero for "01"
         cout << mins << ' ';
         string am_pm = pm ? "p.m." : "a.m.";
         cout << am_pm;
         }
   };
////////////////////////////////////////////////////////////////
class time24
   {
   private:
      int hours;                     //0 to 23
      int minutes;                   //0 to 59
      int seconds;                   //0 to 59
   public:                           //no-arg constructor
      time24() : hours(0), minutes(0), seconds(0)
         {  }
      time24(int h, int m, int s) :  //3-arg constructor
            hours(h), minutes(m), seconds(s)
         {  }
      void display() const           //format: 23:15:01
         {
         if(hours < 10)    cout << '0';
         cout << hours << ':';
         if(minutes < 10)  cout << '0';
         cout << minutes << ':';
         if(seconds < 10)  cout << '0';
         cout << seconds;
         }
      operator time12() const;       //conversion operator
   };
//-------------------------------------------------------------
time24::operator time12() const           //conversion operator
   {
   int hrs24 = hours;
   bool pm = hours < 12 ? false : true;   //find am/pm
                                          //round secs
   int roundMins = seconds < 30 ? minutes : minutes+1;
   if(roundMins == 60)                    //carry mins?
      {
      roundMins=0;
      ++hrs24;
      if(hrs24 == 12 || hrs24 == 24)      //carry hrs?
         pm = (pm==true) ? false : true;  //toggle am/pm
      }
   int hrs12 = (hrs24 < 13) ? hrs24 : hrs24-12;
```