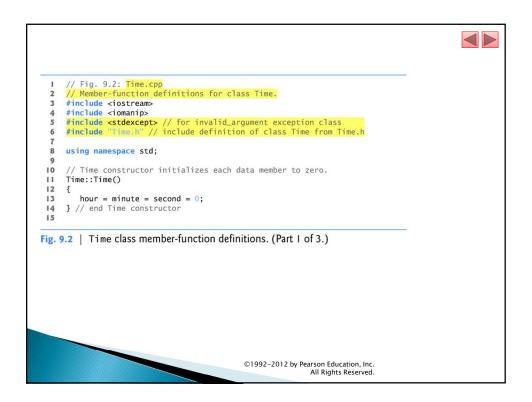




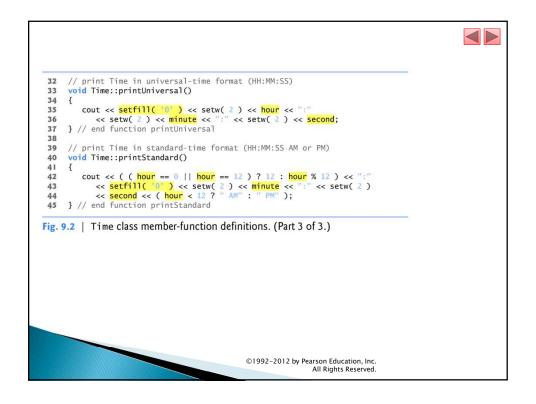
## 9.2 Time Class Case Study

▶ Our first example (Figs. 9.1–9.3) creates class **Time** and a driver program that tests the class.

```
// Fig. 9.1: Time.h
// Time class definition.
// Member functions are defined in Time.cpp
         // prevent multiple inclusions of header #ifndef TIME_H
        // Time class definition class Time
  10
  11
         public:
  12
              Time(); // constructor
  13
              void setTime( int, int ); // set hour, minute and second
void printUniversal(); // print time in universal-time format
void printStandard(); // print time in standard-time format
  15
  16
        int hour; // 0 - 23 (24-hour clock format)
int minute; // 0 - 59
int second; // 0 - 59
}; // end class Time
  18
  19
 20
 21
 22
         #endif
 23
Fig. 9.1 | Time class definition.
                                                                            ©1992-2012 by Pearson Education, Inc.
                                                                                                    All Rights Reserved.
```



```
// set new Time value using universal time
void Time::setTime( int h, int m, int s )
 17
 18
           // validate hour, minute and second if ( ( h >= 0 && h < 24 ) && ( m >= 0 && m < 60 ) && ( s >= 0 && s < 60 ) )
 19
 20
 21
 22
               hour = h;
minute = m;
second = s;
 23
 24
 25
 26
           } // end if
           else
 27
               throw invalid_argument(
 28
                                                 second was out of range" );
 30 } // end function setTime
 31
Fig. 9.2 | Time class member-function definitions. (Part 2 of 3.)
                                                             ©1992-2012 by Pearson Education, Inc.
                                                                                 All Rights Reserved.
```





## 9.2 Time Class Case Study (cont.)

Once class Time has been defined, it can be used as a type in object, array, pointer and reference declarations as follows:

```
Time sunset; // object of type Time
Time arrayOfTimes[ 5 ]; // array of 5 Time objects
Time &dinnerTime = sunset; // reference to a Time object
Time *timePtr = &dinnerTime; // pointer to a Time object
```

```
// Fig. 9.3: fig09_03.cpp

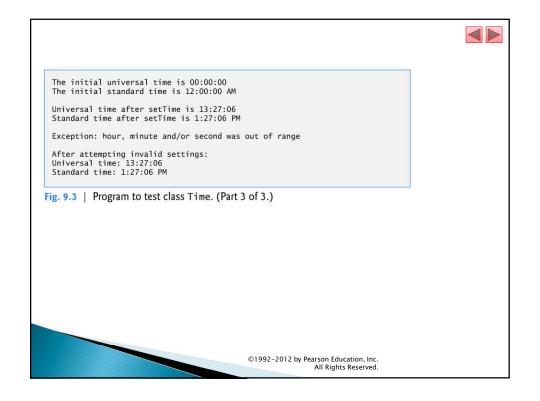
// Program to test class Time.

// NOTE: This file must be compiled with Time.cpp.

#include <iostream>

#include "Time.h" // include definition of class T
                                         include definition of class Time from Time.h
        using namespace std;
        int main()
             Time t; // instantiate object t of class Time
  10
  11
  12
             // output Time object t's initial values
             cout << "The initial universal time is ";
t.printUniversal(); // 00:00:00
cout << "\nThe initial standard time is ";</pre>
  13
 14
15
             t.printStandard(); // 12:00:00 AM
  17
             t.setTime( 13, 27, 6 ); // change time
  18
Fig. 9.3 | Program to test class Time. (Part 1 of 3.)
                                                                      ©1992-2012 by Pearson Education, Inc.
All Rights Reserved.
```

```
// output Time object t's new values
21
 22
         t.printUniversal(); // 13:27:06
         cout << "\nStandard time after se
t.printStandard(); // 1:27:06 PM
 23
24
 25
 26
         // attempt to set the time with invalid values
27
28
         {
 29
            t.setTime( 99, 99, 99 ); // all values out of range
         } // end try
catch ( invalid_argument &e )
{
 30
 31
 32
 33
             cout << "Exception: " << e.what() << endl << endl;</pre>
34
35
         } // end catch
         36
 37
38
         t.printUniversal(); // 00:00:00
 39
    cout << "\nStandard time: ";
t.printStandard(); // 12:00:00 AM
cout << endl;
} // end main</pre>
 40
 41
 42
Fig. 9.3 | Program to test class Time. (Part 2 of 3.)
                                                  ©1992-2012 by Pearson Education, Inc.
                                                                  All Rights Reserved.
```



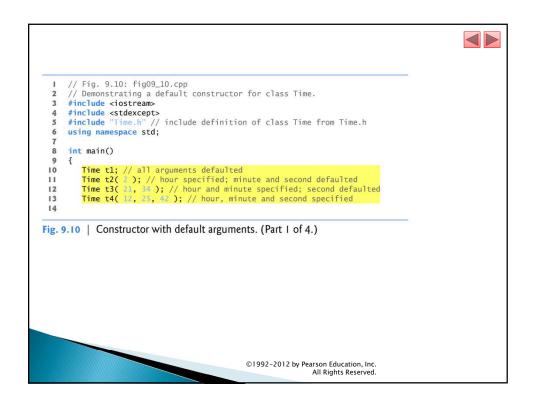


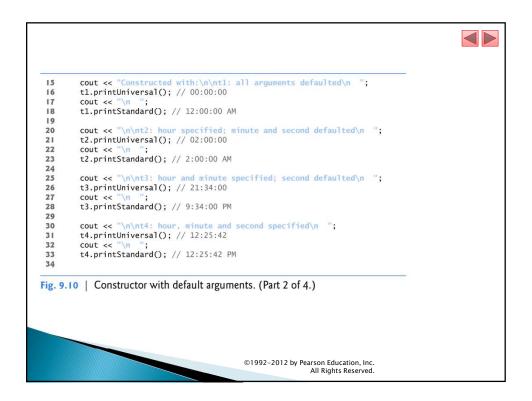
## 9.6 Time Class Case Study: Constructors with Default Arguments

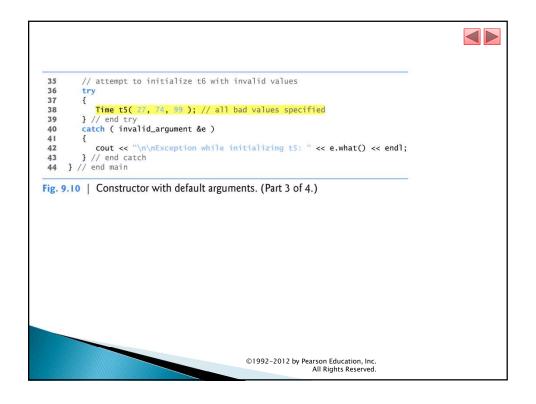
- Like other functions, constructors can specify default arguments.
- The default arguments to the constructor ensure that, even if no values are provided in a constructor call, the constructor still initializes the data.
- A constructor that defaults all its arguments is also a default constructor—i.e., a constructor that can be invoked with no arguments.
- There can be at most one default constructor per class.

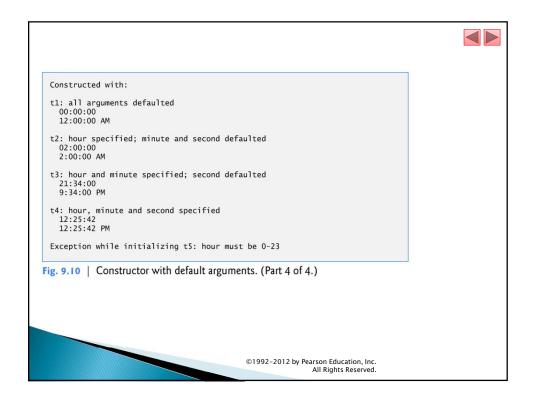
```
// Fig. 9.8: Time.h
       // Time class containing a constructor with default arguments.
       // Member functions defined in Time.cpp.
      // prevent multiple inclusions of header
      #define TIME H
       // Time abstract data type definition
  10
      class Time
 11
      public:
 12
  13
           Time( int = 0, int = 0, int = 0 ); // default constructor
 14
 15
           // set functions
           void setTime( int, int, int ); // set hour, minute, second
void setHour( int ); // set hour (after validation)
void setMinute( int ); // set minute (after validation)
void setSecond( int ); // set second (after validation)
  17
 18
Fig. 9.8 | Time class containing a constructor with default
arguments. (Part I of 2.)
                                                             ©1992-2012 by Pearson Education, Inc.
All Rights Reserved.
```

```
// Fig. 9.9: Time.cpp
// Member-function definitions for class Time.
       #include <iostream>
#include <iomanip>
       #include <stdexcept>
#include "Time.h" // include definition of class Time from Time.h
       using namespace std;
       // Time constructor initializes each data member to zero
 10
       Time::Time( int hour, int minute, int second )
 11
 12
          setTime( hour, minute, second ); // validate and set time
 13
      } // end Time constructor
 15
      // set new Time value using universal time
       void Time::setTime( int h, int m, int s )
 16
 17
      {
      setHour( h ); // set private field hour
setMinute( m ); // set private field minute
setSecond( s ); // set private field second
} // end function setTime
 18
 19
 20
 22
Fig. 9.9 | Time class member-function definitions including a
constructor that takes arguments. (Part 1 of 4.)
                                                         ©1992-2012 by Pearson Education, Inc.
                                                                           All Rights Reserved.
```











## 9.7 Destructors

- ▶ The name of the destructor for a class is the tilde character (~) followed by the class name.
- Often referred to with the abbreviation "dtor" in the literature.
- Called implicitly when an object is destroyed.
- ▶ The destructor itself does not actually release the object's memory—it performs termination housekeeping before the object's memory is reclaimed, so the memory may be reused to hold new objects.
- Receives no parameters and returns no value.
- May not specify a return type—not even **void**.
- A class may have only one destructor.
- A destructor must be public.
- If you do not explicitly provide a destructor, the compiler creates an "empty" destructor.