7.5 Using the this Pointer

• **this** pointer
  – Allows objects to access their own address
  – Not part of the object itself
  – Implicit first argument on member function call to the object
  – Implicitly reference member data and functions
  – The type of the **this** pointer depends upon the type of the object and whether the member function using **this** is **const**
    – In a non-**const** member function of **Employee**, **this** has type `Employee* const`
      • Constant pointer to an **Employee** object
    – In a **const** member function of **Employee**, this has type `const Employee* const`
      • Constant pointer to a constant **Employee** object
7.5 Using the this Pointer

- **Examples using this**
  - For a member function print data member \( x \), either
    \[
    \text{this->} x
    \]
    or
    \[
    (\ast\text{this}).x
    \]

- **Cascaded member function calls**
  - Function returns a reference pointer to the same object
    \[
    \{ \text{return } \ast\text{this;} \}\]
  - Other functions can operate on that pointer
  - Functions that do not return references must be called last
7.5 Using the this Pointer

- Example of cascaded member function calls
  - Member functions `setHour`, `setMinute`, and `setSecond` all return `*this` (reference to an object)
  - For object `t`, consider
    ```cpp
    t.setHour(1).setMinute(2).setSecond(3);
    ```
    - Executes `t.setHour(1)`, returns `*this` (reference to object) and the expression becomes
      ```cpp
      t.setMinute(2).setSecond(3);
      ```
    - Executes `t.setMinute(2)`, returns reference and becomes
      ```cpp
      t.setSecond(3);
      ```
    - Executes `t.setSecond(3)`, returns reference and becomes
      ```cpp
      t;
      ```
    - Has no effect
```cpp
// Fig. 7.7: fig07_07.cpp
// Using the this pointer to refer to object members.
#include <iostream>

using std::cout;
using std::endl;

class Test {
    public:
        Test( int = 0 );       // default constructor
        void print() const;
    private:
        int x;
    }

    Test::Test( int a ) { x = a; }  // constructor

    void Test::print() const // ( ) around *this
    { 
        cout << "        x = " << x 
        << "\n this->x = " << this->x 
        << "\n(*this).x = " << ( *this ).x << endl;
    }

    int main()
    { 
        Test testObject( 12 );
        testObject.print();
        return 0;
    }
```

Printing `x` directly.

Print `x` using the arrow `->` operator off the `this` pointer.

Printing `x` using the dot `(.)` operator. Parenthesis required because dot operator has higher precedence than `*`. Without, interpreted incorrectly as `*(this.x)`. 
All three methods have the same result.
1 // Fig. 7.8: time6.h
2 // Cascading member function calls.
3
4 // Declaration of class Time.
5 // Member functions defined in time6.cpp
6 #ifndef TIME6_H
7 #define TIME6_H
8
9 class Time {
10 public:
11     Time( int = 0, int = 0, int = 0 ); // default constructor
12
13     // set functions
14     Time& setTime( int, int, int ); // set hour, minute, second
15     Time& setHour( int );   // set hour
16     Time& setMinute( int ); // set minute
17     Time& setSecond( int );  // set second
18
19     // get functions (normally declared const)
20     int getHour() const;    // return hour
21     int getMinute() const;  // return minute
22     int getSecond() const;   // return second
23
24     // print functions (normally declared const)
25     void printMilitary() const; // print military time
26     void printStandard() const; // print standard time
27 private:
28     int hour;       // 0 - 23
29     int minute;    // 0 - 59
30     int second;   // 0 - 59
31     
32 #endif

Notice the Time& - function returns a reference to a Time object. Specify object in function definition.
// Fig. 7.8: time.cpp
// Member function definitions for Time class.
#include <iostream>

using std::cout;

#include "time6.h"

// Constructor function to initialize private data.
// Calls member function setTime to set variables.
// Default values are 0 (see class definition).
Time::Time( int hr, int min, int sec )
{ setTime( hr, min, sec ); }

// Set the values of hour, minute, and second.
Time& Time::setTime( int h, int m, int s )
{
    setHour( h );
    setMinute( m );
    setSecond( s );
    return *this;  // enables cascading
}

// Set the hour value
Time& Time::setHour( int h )
{
    hour = ( h >= 0 && h < 24 ) ? h : 0;
    return *this;  // enables cascading
}
// Set the minute value
Time& Time::setMinute( int m )
{
    minute = ( m >= 0 && m < 60 ) ? m : 0;
    return *this; // enables cascading
}

// Set the second value
Time& Time::setSecond( int s )
{
    second = ( s >= 0 && s < 60 ) ? s : 0;
    return *this; // enables cascading
}

// Get the hour value
int Time::getHour() const { return hour; }

// Get the minute value
int Time::getMinute() const { return minute; }

// Get the second value
int Time::getSecond() const { return second; }

// Display military format time: HH:MM
void Time::printMilitary() const
{
    cout << ( hour < 10 ? "0" : "" ) << hour << "":";
    << ( minute < 10 ? "0" : "" ) << minute;
// Display standard format time: HH:MM:SS AM (or PM)
void Time::printStandard() const
{
    cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 ) << " : " << ( minute < 10 ? "0" : "" ) << minute
    << " : " << ( second < 10 ? "0" : "" ) << second
    << ( hour < 12 ? " AM" : " PM" );
}

// Fig. 7.8: fig07_08.cpp
// Cascading member function calls together
// with the this pointer
#include <iostream>
using std::cout;
using std::endl;

#include "time6.h"

int main()
{
    Time t;

    t.setHour( 18 ).setMinute( 30 ).setSecond( 22 );
    cout << "Military time: ";
    t.printMilitary();
    cout << " Standard time: ";
    t.printStandard();

    cout << " New standard time: ";
    t.setTime( 20, 20, 20 ).printStandard();
cout << endl;

return 0;
}

Military time: 18:30
Standard time: 6:30:22 PM

New standard time: 8:20:20 PM