BİL-142 Bilgisayar Programlama II (C/C++)

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Konular

- Giriş
- Örnek: Time Class
- Class Scope ve Class Üyelerine Erişim
- Interface ve Implementation
- Erişim Fonksiyonları
- Default Parametrelerle Constructor
- Destructors
- Memberwise Assignment

Giriş

- Bir class tüm fonksiyonlarını ve değişkenlerini tanımlar.
- Bir class başka class'ların türetilmesi için kullanılabilir.
- Bir class ait fonksiyonlar farklı sayıda ve türe parametrelerle oluşturulabilir.
- Bir class üyelerine erişim kısıtlamaları konulabilir.

3

Konular

- Giriş
- Örnek: Time Class
- Class Scope ve Class Üyelerine Erişim
- Interface ve Implementation
- Erişim Fonksiyonları
- Default Parametrelerle Constructor
- Destructors
- Memberwise Assignment

Örnek: Time Class

```
1  // Fig. 9.1: Time.h
2  // Declaration of class Time.
3  // Member functions are defined in Time.cpp
4
5  // prevent multiple inclusions of header file
6  #ifndef TIME_H
7  #define TIME_H
8
9  // Time class definition
10 class Time
11 {
12 public:
13    Time(); // constructor
14    void setTime( int, int, int ); // set hour, minute and second
15    void printUniversal(); // print time in universal-time format
16    void printStandard(); // print time in standard-time format
17 private:
18    int hour; // 0 - 23 (24-hour clock format)
19    int minute; // 0 - 59
20    int second; // 0 - 59
21 }; // end class Time
22
23 #endif
```

Header file'ın birden fazla kullanılmasını önler

5

Örnek: Time Class

```
1  // Fig. 9.2: Time.cpp
2  // Member-function definitions for class Time.
3  #include <iostream>
4  using std::cout;
5
6  #include <iomanip>
7  using std::setfill;
8  using std::setw;
9
10 #include "Time.h" // include definition of class Time from Time.h
11
12 // Time constructor initializes each data member to zero.
13 // Ensures all Time objects start in a consistent state.
14 Time::Time()
15 {
16    hour = minute = second = 0;
17 } // end Time constructor
18
19 // set new Time value using universal time; ensure that
20 // the data remains consistent by setting invalid values to zero
21 void Time::setTime( int h, int m, int s )
22 {
23    hour = ( h >= 0 && h < 24 ) ? h : 0; // validate hour
24    minute = ( m >= 0 && m < 60 ) ? m : 0; // validate minute
25    second = ( s >= 0 && s < 60 ) ? s : 0; // validate second
26 } // end function setTime
27
```

Data üyeleri 0 yapıldı

Public fonksiyon saati ayarlar

6

Örnek: Time Class

```
28 // print Time in universal-time format (HH:MM:SS)
29 void Time::printUniversal()
30 {
31    cout << setfill( '0' ) << setw( 2 ) << hour << ":"
32       << setw( 2 ) << minute << ":" << setw( 2 ) << second;
33 } // end function printUniversal
34
35 // print Time in standard-time format (HH:MM:SS AM or PM)
36 void Time::printStandard()
37 {
38    cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 ) << ":"
39       << setfill( '0' ) << setw( 2 ) << minute << ":" << setw( 2 )
40       << second << ( hour < 12 ? " AM" : " PM" );
41 } // end function printStandard
42
```

Rakamların sol kısmı 0 yapılır (sticky setting)

`setw()` nonsticky setting

7

Örnek: Time Class

```
1  // Fig. 9.3: fig09_03.cpp
2  // Program to test class Time.
3  // NOTE: This file must be compiled with Time.cpp.
4  #include <iostream>
5  using std::cout;
6  using std::endl;
7
8  #include "Time.h" // include definition of class Time from Time.h
9
10 int main()
11 {
12    Time t; // instantiate object t of class Time
13
14    // output Time object t's initial values
15    cout << "The initial universal time is ";
16    t.printUniversal(); // 00:00:00
17    cout << "\nThe initial standard time is ";
18    t.printStandard(); // 12:00:00 AM
19
20    t.setTime( 13, 27, 6 ); // change time
21
```

8

Örnek: Time Class

```
22    // output Time object t's new values
23    cout << "\n\nUniversal time after setTime is ";
24    t.printUniversal(); // 13:27:06
25    cout << "\nStandard time after setTime is ";
26    t.printStandard(); // 1:27:06 PM
27
28    t.setTime( 99, 99, 99 ); // attempt invalid settings
29
30    // output t's values after specifying invalid values
31    cout << "\n\nAfter attempting invalid settings:"
32       << "\nUniversal time: ";
33    t.printUniversal(); // 00:00:00
34    cout << "\nStandard time: ";
35    t.printStandard(); // 12:00:00 AM
36    cout << endl;
37
38    system("PAUSE");
39    return 0;
40 } // end main
```

```
c:\m.ali\deneme\Debug\deneme.exe
The initial universal time is 00:00:00
The initial standard time is 12:00:00 AM

Universal time after setTime is 13:27:06
Standard time after setTime is 1:27:06 PM

After attempting invalid settings:
Universal time: 00:00:00
Standard time: 12:00:00 AM
Press any key to continue . . .
```

9

Örnek: Time Class

- **`time`** class içinde **`setTime`** fonksiyonu **`public`** tanımlanmıştır ve **`private`** tanımlanmış **`hour`**, **`minute`** ve **`second`** değişkenlerine değer atar.
- **`setTime`** girilen değerlerin doğruluğunuda kontrol eder.
- **`printUniversal`** ve **`printStandart`** fonksiyonları **`public`** tanımlanmıştır ve zaman bilgisini belirlenen formatlarda ekrana yazar.
- Time class aşağıdaki gibi kullnaılabilir.

```
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
```

10

Konular

- Giriş
- Örnek: Time Class
- Class Scope ve Class Üyelerine Erişim
- Interface ve Implementation
- Erişim Fonksiyonları
- Default Parametrelerle Constructor
- Destructors
- Memberwise Assignment

Class Scope ve Class Üyelerine Erişim

- Bir class içinde tanımlı data üyeleri (değişkenler) ve fonksiyon üyeleri class scope çalışır.
- Üye olmayan fonksiyonlar file scope çalışır.
- Üye fonksiyonlar overload yapılabilir. Bir fonksiyonu overload yapmak için prototype tanımlamasının yapılması gerekir.
- Class içinde tanımlı değişkenler calss scapo çalışır. Bir blokta aynı adlı değişken varsa blok içindeki önceliklidir. Scope önceliklendirmesi blok, fonksiyon, class, file şeklindedir.

12

Class Scope ve Class Üyelerine Erişim

- Bir önceki scope seviyesine :: scope resolution operatörü ile ulaşılır.
- Bir nesnenin üyelerine . seçme operatörü ile ulaşılır.
- Bir pointer ile nesne üyelerine -> operatörü ile ulaşılır.

13

Class Scope ve Class Üyelerine Erişim

```
1  // Fig. 9.4: fig09_04.cpp
2  // Demonstrating the class member access operators . and ->
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  // class Count definition
8  class Count
9  {
10 public: // public data is dangerous
11    // sets the value of private data member x
12    void setX( int value )
13    {
14       x = value;
15    } // end function setX
16
17    // prints the value of private data member x
18    void print()
19    {
20       cout << x << endl;
21    } // end function print
22
23 private:
24    int x;
25 }; // end class Count
26
```

14

Class Scope ve Class Üyelerine Erişim

```
27 int main()
28 {
29    Count counter; // create counter object
30    Count *counterPtr = &counter; // create pointer to counter
31    Count &counterRef = counter; // create reference to counter
32
33    cout << "Set x to 1 and print using the object's name: ";
34    counter.setX( 1 ); // set data member x to 1
35    counter.print(); // call member function print
36
37    cout << "Set x to 2 and print using a reference to an object: ";
38    counterRef.setX( 2 ); // set data member x to 2
39    counterRef.print(); // call member function print
40
41    cout << "Set x to 3 and print using a pointer to an object: ";
42    counterPtr->setX( 3 ); // set data member x to 3
43    counterPtr->print(); // call member function print
44
45    system("PAUSE");
46    return 0;
47 } // end main
48
```

```
c:\m.ali\deneme\Debug\deneme.exe
Set x to 1 and print using the object's name: 1
Set x to 2 and print using a reference to an object: 2
Set x to 3 and print using a pointer to an object: 3
Press any key to continue . . .
```

15

Konular

- Giriş
- Örnek: Time Class
- Class Scope ve Class Üyelerine Erişim
- Interface ve Implementation
- Erişim Fonksiyonları
- Default Parametrelerle Constructor
- Destructors
- Memberwise Assignment

Interface ve Implementation

- Bir class'ın tanımlaması (interface - .h) ile class üyelerinin kaynak kodunun (implementation - .cpp) ayrı dosyalarda oluşturulabilir.
- Kullanıcılar bir class'ı interface kısmı ile kullanabilir.
- Implementation kısmı değişmiş olsa bile (interface orijinal kalmak kaydıyla) kullanıcılar bundan etkilenmez.

17

Konular

- Giriş
- Örnek: Time Class
- Class Scope ve Class Üyelerine Erişim
- Interface ve Implementation
- Erişim Fonksiyonları
- Default Parametrelerle Constructor
- Destructors
- Memberwise Assignment

Erişim Fonksiyonları

- Erişim fonksiyonları bir class içindeki bilgileri okumak ve ekranda görüntülemek için kullanılır.
- Predicate fonksiyonalar ise bir durum hakkında bilgi almak için kullanılır (**`isEmpty`**, **`isFull`**).

19

Erişim Fonksiyonları

```
1  // Fig. 9.5: SalesPerson.h
2  // SalesPerson class definition.
3  // Member functions defined in SalesPerson.cpp.
4  #ifndef SALESP_H
5  #define SALESP_H
6
7  class SalesPerson
8  {
9  public:
10    SalesPerson(); // constructor
11    void getSalesFromUser(); // input sales from keyboard
12    void setSales( int, double ); // set sales for a specific month
13    void printAnnualSales(); // summarize and print sales
14 private:
15    double totalAnnualSales(); // prototype for utility function
16    double sales[ 12 ]; // 12 monthly sales figures
17 }; // end class SalesPerson
18
19 #endif
```

20

Erişim Fonksiyonları

```
1  // Fig. 9.6: SalesPerson.cpp
2  // Member functions for class SalesPerson.
3  #include <iostream>
4  using std::cout;
5  using std::cin;
6  using std::endl;
7  using std::fixed;
8
9  #include <iomanip>
10 using std::setprecision;
11
12 #include "SalesPerson.h" // include SalesPerson class definition
13
14 // initialize elements of array sales to 0.0
15 SalesPerson::SalesPerson()
16 {
17    for ( int i = 0; i < 12; i++ )
18       sales[ i ] = 0.0;
19 } // end SalesPerson constructor
20
```

21

Erişim Fonksiyonları

```
21 // get 12 sales figures from the user at the keyboard
22 void SalesPerson::getSalesFromUser()
23 {
24    double salesFigure;
25
26    for ( int i = 1; i <= 12; i++ )
27    {
28       cout << "Enter sales amount for month " << i << ": ";
29       cin >> salesFigure;
30       setSales( i, salesFigure );
31    } // end for
32 } // end function getSalesFromUser
33
34 // set one of the 12 monthly sales figures; function subtracts
35 // one from month value for proper subscript in sales array
36 void SalesPerson::setSales( int month, double amount )
37 {
38    // test for valid month and amount values
39    if ( month >= 1 && month <= 12 && amount > 0 )
40       sales[ month - 1 ] = amount; // adjust for subscripts 0-11
41    else // invalid month or amount value
42       cout << "Invalid month or sales figure" << endl;
43 } // end function setSales
44
```

22

Erişim Fonksiyonları

```
45  // print total annual sales (with the help of utility function)
46  void SalesPerson::printAnnualSales()
47  {
48     cout << setprecision( 2 ) << fixed
49        << "\nThe total annual sales are: $"
50        << totalAnnualSales() << endl; // call utility function
51  } // end function printAnnualSales
52
53  // private utility function to total annual sales
54  double SalesPerson::totalAnnualSales()
55  {
56     double total = 0.0; // initialize total
57
58     for ( int i = 0; i < 12; i++ ) // summarize sales results
59        total += sales[ i ]; // add month i sales to total
60
61     return total;
62  } // end function totalAnnualSales
63
```


Erişim Fonksiyonları

```
1  // Fig. 9.7: fig09_07.cpp
2  // Demonstrating a utility function.
3  // Compile this program with SalesPerson.cpp
4
5  // include SalesPerson class definition from SalesPerson.h
6  #include "SalesPerson.h"
7
8  int main()
9  {
10    SalesPerson s; // create SalesPerson object s
11
12    s.getSalesFromUser(); // note simple sequential code;
13    s.printAnnualSales(); // no control statements in main
14
15    system("PAUSE");
16    return 0;
17 } // end main
18
```

```
c:\m.ali\deneme\Debug\deneme.exe
Enter sales amount for month 1: 1232.13
Enter sales amount for month 2: 3234.45
Enter sales amount for month 3: 5646.56
Enter sales amount for month 4: 6546.78
Enter sales amount for month 5: 7868.80
Enter sales amount for month 6: 8769.97
Enter sales amount for month 7: 3425.54
Enter sales amount for month 8: 5364.87
Enter sales amount for month 9: 4759.08
Enter sales amount for month 10: 1212.23
Enter sales amount for month 11: 2143.32
Enter sales amount for month 12: 3461.21

The total annual sales are: $53664.94
Press any key to continue . . .
```

Konular

- Giriş
- Örnek: Time Class
- Class Scope ve Class Üyelerine Erişim
- Interface ve Implementation
- Erişim Fonksiyonları
- Default Parametrelerle Constructor
- Destructors
- Memberwise Assignment

Default Parametrelerle Constructor

- Bir sınıf constructor'ı ile değişkenlerine başlangıç değeri atayabilir.
- Böyle bir sınıftan yeni bir örnek oluşturulduğunda constructor'a parametre verilmezse default değerleri alır.
- Böyle bir sınıftan yeni bir örnek oluşturulurken constructor'dakinden az parametre kullanılırsa soldan itibaren atama yapılır.

26

Default Parametrelerle Constructor

```
1  // Fig. 9.8: Time.h
2  // Declaration of class Time.
3  // Member functions defined in Time.cpp.
4
5  // prevent multiple inclusions of header file
6  #ifndef TIME_H
7  #define TIME_H
8
9  // Time abstract data type definition
10 class Time
11 {
12 public:
13    Time( int = 0, int = 0, int = 0 ); // default constructor
14
15    // set functions
16    void setTime( int, int, int ); // set hour, minute, second
17    void setHour( int ); // set hour (after validation)
18    void setMinute( int ); // set minute (after validation)
19    void setSecond( int ); // set second (after validation)
20
21    // get functions
22    int getHour(); // return hour
23    int getMinute(); // return minute
24    int getSecond(); // return second
```

27

Default Parametrelerle Constructor

```
25
26    void printUniversal(); // output time in universal-time format
27    void printStandard(); // output time in standard-time format
28 private:
29    int hour; // 0 - 23 (24-hour clock format)
30    int minute; // 0 - 59
31    int second; // 0 - 59
32 }; // end class Time
33
34 #endif
35
```

28

Default Parametrelerle Constructor

```
1  // Fig. 9.9: Time.cpp
2  // Member-function definitions for class Time.
3  #include <iostream>
4  using std::cout;
5
6  #include <iomanip>
7  using std::setfill;
8  using std::setw;
9
10 #include "Time.h" // include definition of class Time from Time.h
11
12 // Time constructor initializes each data member to zero;
13 // ensures that Time objects start in a consistent state
14 Time::Time( int hr, int min, int sec )
15 {
16    setTime( hr, min, sec ); // validate and set time
17 } // end Time constructor
18
19 // set new Time value using universal time; ensure that
20 // the data remains consistent by setting invalid values to zero
21 void Time::setTime( int h, int m, int s )
22 {
23    setHour( h ); // set private field hour
24    setMinute( m ); // set private field minute
25    setSecond( s ); // set private field second
26 } // end function setTime
27
```

29

Default Parametrelerle Constructor

```
28 // set hour value
29 void Time::setHour( int h )
30 {
31    hour = ( h >= 0 && h < 24 ) ? h : 0; // validate hour
32 } // end function setHour
33
34 // set minute value
35 void Time::setMinute( int m )
36 {
37    minute = ( m >= 0 && m < 60 ) ? m : 0; // validate minute
38 } // end function setMinute
39
40 // set second value
41 void Time::setSecond( int s )
42 {
43    second = ( s >= 0 && s < 60 ) ? s : 0; // validate second
44 } // end function setSecond
45
46 // return hour value
47 int Time::getHour()
48 {
49    return hour;
50 } // end function getHour
51
```

30

Default Parametrelerle Constructor

```
1  // Fig. 9.10: fig09_10.cpp
2  // Demonstrating a default constructor for class Time.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include "Time.h" // include definition of class Time from Time.h
8
9  int main()
10 {
11    Time t1; // all arguments defaulted
12    Time t2( 2 ); // hour specified; minute and second defaulted
13    Time t3( 21, 34 ); // hour and minute specified; second defaulted
14    Time t4( 12, 25, 42 ); // hour, minute and second specified
15    Time t5( 27, 74, 99 ); // all bad values specified
16
17    cout << "Constructed with:\n\nt1: all arguments defaulted\n  ";
18    t1.printUniversal(); // 00:00:00
19    cout << "\n  ";
20    t1.printStandard(); // 12:00:00 AM
21
22    cout << "\n\nt2: hour specified; minute and second defaulted\n  ";
23    t2.printUniversal(); // 02:00:00
24    cout << "\n  ";
25    t2.printStandard(); // 2:00:00 AM
```


Default Parametrelerle Constructor

```
26
27    cout << "\n\nt3: hour and minute specified; second defaulted\n  ";
28    t3.printUniversal(); // 21:34:00
29    cout << "\n  ";
30    t3.printStandard(); // 9:34:00 PM
31
32    cout << "\n\nt4: hour, minute and second specified\n  ";
33    t4.printUniversal(); // 12:25:42
34    cout << "\n  ";
35    t4.printStandard(); // 12:25:42 PM
36
37    cout << "\n\nt5: all invalid values specified\n  ";
38    t5.printUniversal(); // 00:00:00
39    cout << "\n  ";
40    t5.printStandard(); // 12:00:00 AM
41    cout << endl;
42
43    system("PAUSE");
44    return 0;
45 } // end main
46
```

```
c:\m.ali\ddd\Debug\ddd.exe
Constructed with:

t1: all arguments defaulted
  00:00:00
  12:00:00 AM

t2: hour specified; minute and second defaulted
  02:00:00
  2:00:00 AM

t3: hour and minute specified; second defaulted
  21:34:00
  9:34:00 PM

t4: hour, minute and second specified
  12:25:42
  12:25:42 PM

t5: all invalid values specified
  00:00:00
  12:00:00 AM
Press any key to continue . . .
```

Konular

- Giriş
- Örnek: Time Class
- Class Scope ve Class Üyelerine Erişim
- Interface ve Implementation
- Erişim Fonksiyonları
- Default Parametrelerle Constructor
- Destructors
- Memberwise Assignment

Destructors

- Destructor bir class'ın adıyla aynı ada sahip ve önünde (~ - tilde) karakteri olan fopnksiyon üyesidir.
- Destructor ile constructor birbirinin tersidir.
- Bir class destructor'ı nesne'nin kullanımı bittiğinde dolaylı olarak çağırılır.
- Bir destructor, nesnenin ait olduğu scope içindeki çalışması bittiğinde otomatik olarak çalışır.
- Destructor, parametre almaz ve geri döndürmez, overload yapılamaz.

34

Destructors

- Bir class için destructor tanımlanmasa bile compiler otomatik olarak oluşturur.
- Constructor bir program çalışmaya başladığında main de dahil tüm fonksiyonlardan önce çalışır.
- Constructor ve destructor'lar bir nesnenin scope'una girince ve çıkınca otomatik olarak çalıştırılır.

35

Destructors

```
1  // Fig. 9.11: CreateAndDestroy.h
2  // Definition of class CreateAndDestroy.
3  // Member functions defined in CreateAndDestroy.cpp.
4  #include <string>
5  using std::string;
6
7  #ifndef CREATE_H
8  #define CREATE_H
9
10 class CreateAndDestroy
11 {
12 public:
13    CreateAndDestroy( int, string ); // constructor
14    ~CreateAndDestroy(); // destructor
15 private:
16    int objectID; // ID number for object
17    string message; // message describing object
18 }; // end class CreateAndDestroy
19
20 #endif
```

Her nesneyi diğerlerinden ayrırırlar

36

Destructors

```
// Fig. 9.12: CreateAndDestroy.cpp
// Member-function definitions for class CreateAndDestroy.
#include <iostream>
using std::cout;
using std::endl;

#include "CreateAndDestroy.h" // include CreateAndDestroy class definition

// constructor
CreateAndDestroy::CreateAndDestroy( int ID, string messageString )
{
   objectID = ID; // set object's ID number
   message = messageString; // set object's descriptive message

   cout << "Object " << objectID << "   constructor runs    "
      << message << endl;
} // end CreateAndDestroy constructor

// destructor
CreateAndDestroy::~CreateAndDestroy()
{
   // output newline for certain objects; helps readability
   cout << ( objectID == 1 || objectID == 6 ? "\n" : "" );

   cout << "Object " << objectID << "   destructor runs    "
      << message << endl;
} // end ~CreateAndDestroy destructor
```

Nesnenin ID değerine göre destructor çalışır

37

Destructors

```
// Fig. 9.13: fig09_13.cpp
// Demonstrating the order in which constructors and
// destructors are called.
#include <iostream>
using std::cout;
using std::endl;

#include "CreateAndDestroy.h" // include CreateAndDestroy class definition

void create( void ); // prototype

CreateAndDestroy first( 1, "(global before main)" ); // global object

int main()
{
   cout << "\nMAIN FUNCTION: EXECUTION BEGINS" << endl;
   CreateAndDestroy second( 2, "(local automatic in main)" );
   static CreateAndDestroy third( 3, "(local static in main)" );

   create(); // call function to create objects

   cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
   CreateAndDestroy fourth( 4, "(local automatic in main)" );
   cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
   return 0;
} // end main
```

Global scope'ta tanımlı nesne

3 nesne tanımlanıyor

38

Destructors

```
28  // function to create objects
29  void create( void )
30  {
31     cout << "\nCREATE FUNCTION: EXECUTION BEGINS" << endl;
32     CreateAndDestroy fifth( 5, "(local automatic in create)" );
33     static CreateAndDestroy sixth( 6, "(local static in create)" );
34     CreateAndDestroy seventh( 7, "(local automatic in create)" );
35     cout << "\nCREATE FUNCTION: EXECUTION ENDS" << endl;
36  } // end function create
37
```

```
c:\Documents and Settings\m.ali\My Documents\Visual Studio 2008\Proj
Object 1   constructor runs   (global before main)

MAIN FUNCTION: EXECUTION BEGINS
Object 2   constructor runs   (local automatic in main)
Object 3   constructor runs   (local static in main)

CREATE FUNCTION: EXECUTION BEGINS
Object 5   constructor runs   (local automatic in create)
Object 6   constructor runs   (local static in create)
Object 7   constructor runs   (local automatic in create)

CREATE FUNCTION: EXECUTION ENDS
Object 7   destructor runs    (local automatic in create)
Object 5   destructor runs    (local automatic in create)

MAIN FUNCTION: EXECUTION RESUMES
Object 4   constructor runs   (local automatic in main)

MAIN FUNCTION: EXECUTION ENDS
Press any key to continue . . .
```

39

Konular

- Giriş
- Örnek: Time Class
- Class Scope ve Class Üyelerine Erişim
- Interface ve Implementation
- Erişim Fonksiyonları
- Default Parametrelerle Constructor
- Destructors
- Memberwise Assignment

Memberwise Assignment

- Aynı türdeki nesneler arasında = operatörü ile atama yapılabilir.
- Eşitliğin sağ tarafındaki nesnenin tüm data üyelerinin değerleri sol taraftaki nesnenin data üyelerine aktarılır.
- Memberwise assignment data ğyeleri pointer içeren nesneler problemler oluşturur.

41

Memberwise Assignment

```
1  // Fig. 9.17: Date.h
2  // Declaration of class Date.
3  // Member functions are defined in Date.cpp
4
5  // prevent multiple inclusions of header file
6  #ifndef DATE_H
7  #define DATE_H
8
9  // class Date definition
10 class Date
11 {
12 public:
13    Date( int = 1, int = 1, int = 2000 ); // default constructor
14    void print();
15 private:
16    int month;
17    int day;
18    int year;
19 }; // end class Date
20
21 #endif
```

42

Memberwise Assignment

```
1  // Fig. 9.18: Date.cpp
2  // Member-function definitions for class Date.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include "Date.h" // include definition of class Date from Date.h
8
9  // Date constructor (should do range checking)
10 Date::Date( int m, int d, int y )
11 {
12    month = m;
13    day = d;
14    year = y;
15 } // end constructor Date
16
17 // print Date in the format mm/dd/yyyy
18 void Date::print()
19 {
20    cout << month << '/' << day << '/' << year;
21 } // end function print
```

43

Memberwise Assignment

```
1  // Fig. 9.19: fig09_19.cpp
2  // Demonstrating that class objects can be assigned
3  // to each other using default memberwise assignment.
4  #include <iostream>
5  using std::cout;
6  using std::endl;
7
8  #include "Date.h" // include definition of class Date from Date.h
9
10 int main()
11 {
12    Date date1( 7, 4, 2004 );
13    Date date2; // date2 defaults to 1/1/2000
14
15    cout << "date1 = ";
16    date1.print();
17    cout << "\ndate2 = ";
18    date2.print();
19
20    date2 = date1; // default memberwise assignment
21
22    cout << "\n\nAfter default memberwise assignment, date2 = ";
23    date2.print();
24    cout << endl;
25    return 0;
26 } // end main
```

```
c:\Documents and Settings\m.ali\My Documents\Visual Studio 2008\
date1 = 7/4/2004
date2 = 1/1/2000

After default memberwise assignment, date2 = 7/4/2004
Press any key to continue . . .
```

44