Πανεπιστήμιο Πατρών

Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής

ΟΝΤΟΚΕΝΤΡΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΙΙ (C++)

Κλάσεις και Αφαίρεση Δεδομένων

Outline

6.1 Εισαγωγή

- Αντικειμενοστραφής Προγραμματισμός
  Οντοκεντρικός Προγραμματισμός
  Object-oriented programming (OOP)
  - Εμφωλεύει δεδομένα (χαρακτηριστικά) και συναρτήσεις (συμπεριφορά) σε πακέτα που καλούνται κλάσεις
- Απόκρυψη πληροφορίας
  - Οι κλάσεις (αντικείμενα) επικοινωνούν με καλώς ορισμένες διεπαφές
  - Οι λεπτομέρειες της υλοποίησης βρίσκονται κρυμμένες μέσα στις ίδιες τις κλάσεις
- Τύποι οριζόμενοι από το χρήστη: κλάσεις
  - Δεδομένα (data members)
  - Συναρτήσεις (member functions or methods)
  - Έκφανση-εκτέλεση κλάσης: αντικείμενο

6.2 Ορισμός Δομών-Structure

- Δομές (Structs)
  - Καθορίζουν τύπους που προκύπτουν από τη συνάθροιση μελών άλλων τύπων

```
struct Time {
       int hour;
       int minute;
       int second;
};
```

Structure tag

Structure members

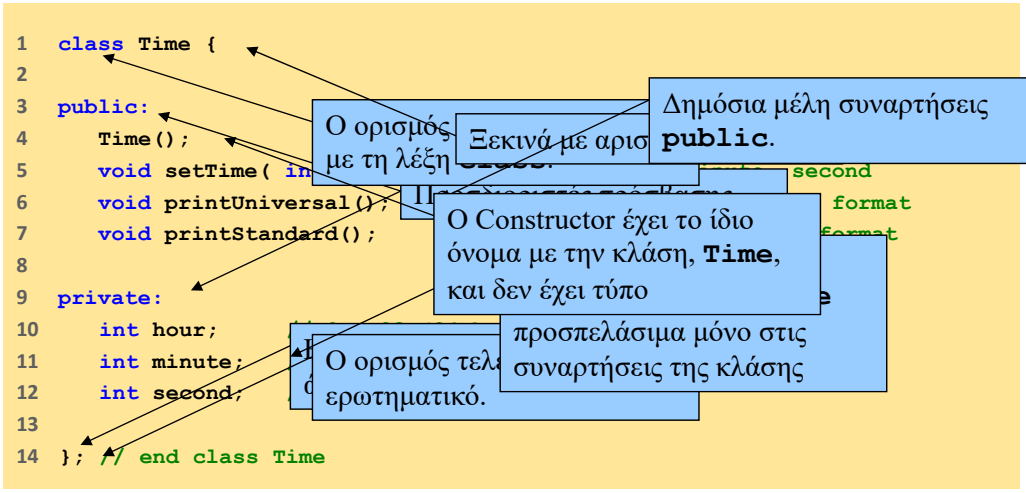
  - Δημιουργεί έναν νέο τύπο δεδομένων που χρησιμοποιείται για να δηλώσει κανείς μεταβλητές
- Ο ορισμός `struct` τελειώνει πάντα με ερωτηματικό
- Στη C++ **struct** και **class** είναι ισοδύναμα. Διαφορά:
  - Struct: default **public** μέλη
  - Class: default **private** μέλη

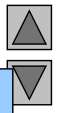
6.5 Υλοποίηση αφηρημένου τύπου **`Time`** με κλάση **`class`**

- Κλάσεις
  - Μοντελοποίηση αντικειμένων
    - Χαρακτηριστικά - Attributes (μέλη δεδομένα)
    - Συμπεριφορές - Behaviors (μέλη συναρτήσεις)
  - Ορίζεται με τη λέξη-κλειδί **`class`**
  - Μέλη συναρτήσεις
    - Μέθοδοι
    - Καλούνται σε απάντηση μηνύματος
- Καθοριστές πρόσβασης σε μέλη
  - **`Δημόσια - public:`**
    - Πρόσβαση σε οποιοδήποτε αντικείμενο της κλάσης εντός περιοχής -scope
  - **`Ιδιωτικό - private:`**
    - Πρόσβαση μόνο από μέλη συναρτήσεις της κλάσης
  - **`Προστατευόμενο - protected:`**
    - Πρόσβαση από παράγωγες κλάσεις

6.5 Υλοποίηση αφηρημένου τύπου `Time` με κλάση `class`

- Συνάρτηση Constructor
  - Ειδικό μέλος συνάρτηση
    - Αρχικοποιεί τα δεδομένα
    - Έχει το ίδιο όνομα με την κλάση
  - Καλείται όταν παράγεται το αντικείμενο
  - Μπορεί να υπάρχουν πολλοί constructors
    - Υπερφόρτωση συναρτήσεων Function
    - Υπάρχει default, καλύπτεται με ορισμό χωρίς arguments
  - Δεν έχει/επιστρέφει κάποιο type (εξ ορισμού void)
  - Public ή private





Outline

Class `Time` definition (1 of 1)

6.5 Υλοποίηση αφηρημένου τύπου `Time` με κλάση `class`

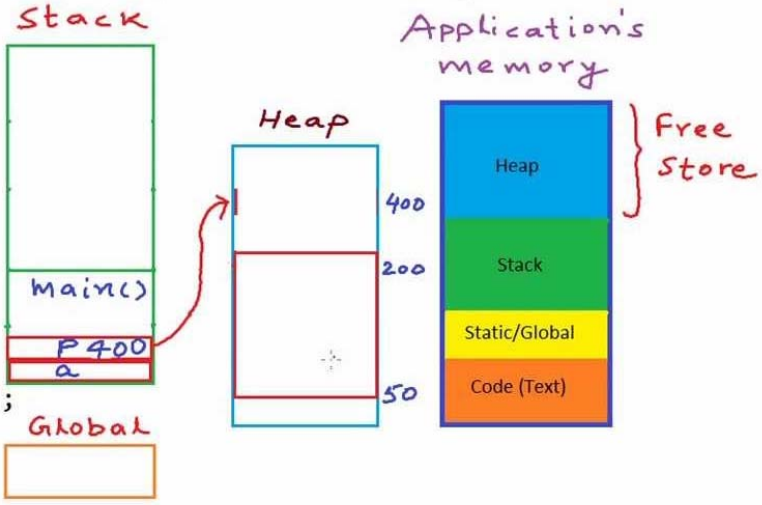
- Αντικείμενα της κλάσης
  - Μετά τον ορισμό της κλάσης
    - Το όνομα κλάσης είναι ένας προσδιοριστής ενός νέου τύπου
  - Example:

Το όνομα κλάσης προσδιορίζει ένα νέο τύπο

```
Time sunset;                  // object of type Time
Time arrayOfTimes[ 5 ];       // array of Time objects
Time *pointerToTime;          // pointer to a Time object
Time &dinnerTime = sunset;    // reference to a Time object
Time noon = Time();           // no use of new!!!
Time noon(12);                // constructor init
```

Stack vs Heap

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
   int a; // goes on stack
   int *p;
   p = (int*)malloc(sizeof(int));
   *p = 10;
   free(p);
 p = (int*)malloc(20*sizeof(int));
}
```

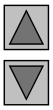


6.5 Υλοποίηση αφηρημένου τύπου `Time` με κλάση `class`

- Μέλη συναρτήσεις εκτός κλάσης
  - Binary scope resolution operator (`::`)
    - Δένει το όνομα μέλους με μια κλάση
    - Προσδιορίζει μοναδικά τις συναρτήσεις συγκεκριμένων κλάσεων
    - Διαφορετικές κλάσεις μπορούν να έχουν συναρτήσεις μέλη με το ίδιο όνομα
  - Μορφή για τον ορισμό συναρτήσεων μελών

```
ReturnType ClassName::MemberFunctionName( ){
  …
}
```

  - Δεν αλλάζει αν η συνάρτηση είναι **`public`** ή **`private`**
- Μέλη συναρτήσεις εντός κλάσης
  - Δε χρειάζονται τελεστή ορισμού scope και όνομα κλάσης

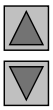


Outline

fig06_03.cpp (1 of 5)

```
1  // Fig. 6.3: fig06_03.cpp
2  // Time class.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  #include <iomanip>
9
10 using std::setfill;
11 using std::setw;
12
13 // Time abstract data type (ADT) definition
14 class Time {
15
16 public:
17    Time();                          // constructor
18    void setTime( int, int, int ); // set hour, minute, second
19    void printUniversal();           // print universal-time format
20    void printStandard();            // print standard-time format
21
```

Κλάση **Time**.



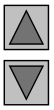
Outline

fig06_03.cpp (2 of 5)

```
22 private:
23    int hour;      // 0 - 23 (24-hour clock format)
24    int minute;    // 0 - 59
25    int second;    // 0 - 59
26
27 }; // end class Time
28
29 // Time constructor initializes each data me
30 // ensures all Time objects start in a consi
31 Time::Time()
32 {
33    hour = minute = second = 0;
34
35 } // end Time constructor
36
37 // set new Time value using universal time, perform validity
38 // checks on the data values and set invalid values to zero
39 void Time::setTime( int h, int m, int s )
40 {
41    hour = ( h >= 0 && h < 24 ) ? h : 0;
42    minute = ( m >= 0 && m < 60 ) ? m : 0;
43    second = ( s >= 0 && s < 60 ) ? s : 0;
44
45 } // end function setTime
46
```

Ο Constructor αρχικοποιεί τα **private** δεδομένα στην τιμή μηδέν (**0**).

Η **public** συνάρτηση ελέγχει τις τιμές των παραμέτρων πριν ορίσει τα **private** δεδομένα.



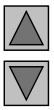
Outline

fig06_03.cpp (3 of 5)

```
47 // print Time in universal format
48 void Time::printUniversal()
49 {
50    cout << setfill( '0' ) << setw( 2 ) << hour << ":"
51         << setw( 2 ) << minute << ":"
52         << setw( 2 ) << second;
53
54 } // end function printUniversal
55
56 // print Time in standard format
57 void Time::printStandard()
58 {
59    cout << ( ( hour == 0 || hour == 12 ) ?
60         << ":" << setfill( '0' ) << setw( 2
61         << ":" << setw( 2 ) << second
62         << ( hour < 12 ? " AM" : " PM" );
63
64 } // end function print
65
66 int main()
67 {
68    Time t;  // instantiate object t of class Time
69
```

Χωρίς παραμέτρους ! (γνωρίζει implicitly ότι σκοπός είναι να εκτυπωθούν μέλη δεδομένα! Έτσι γίνεται περιεκτικότερη η κλήση των συναρτήσεων.

Δηλώνουμε τη μεταβλητή **t** ως αντικείμενο κλάσης **Time**.



Outline

fig06_03.cpp (4 of 5)

```
70    // output Time object t's initial values
71    cout << "The initial universal time is ";
72    t.printUniversal(); // 00:00:00
73
74    cout << "\nThe initial standard time is
75    t.printStandard(); // 12:00:00 AM
76
77    t.setTime( 13, 27, 6 );   // change time
78
79    // output Time object t's new valu
80    cout << "\n\nUniversal time after
81    t.printUniversal();   // 13:27:06
82
83    cout << "\nStandard time after se
84    t.printStandard();    // 1:27:06 P
85
86    t.setTime( 99, 99, 99 );  // attempt invalid settings
87
88    // output t's values after specifying invalid values
89    cout << "\n\nAfter attempting invalid settings:"
90         << "\nUniversal time: ";
91    t.printUniversal();   // 00:00:00
92
```

Καλούμε τη **public** συνάρτηση για εκτύπωση.

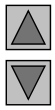
Ορίζουμε τα μέλη δεδομένα χρησιμοποιώντας τη **public** συνάρτηση μέλος.

Προσπάθεια να περάσουμε λάθος τιμές με χρήση της **public** συνάρτησης.

```
93     cout << "\nStandard time: ";
94     t.printStandard();     // 12:00:00 AM
95     cout << endl;
96
97     return 0;
98
99  } // end main
```

```
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
```

Τα μέλη δεδομένων γίνονται μηδέν (0) μετά από τις λάθος τιμές που δόθηκαν.



Outline

fig06_03.cpp (5 of 5)

fig06_03.cpp output (1 of 1)

6.5 Υλοποίηση αφηρημένου τύπου `Time` με κλάση `class`

- Destructors
  - Έχουν το ίδιο όνομα με την κλάση
    - Προηγείται το σημάδι μαθηματικής άρνησης (tilde) (~)
  - Χωρίς παραμέτρους
  - Δε μπορεί να υπερφορτωθεί
  - Public

6.5 Υλοποίηση αφηρημένου τύπου `Time` με κλάση `class`

- Πλεονεκτήματα χρήσης κλάσεων
  - Απλοποιούν τον προγραμματισμό
  - Διεπαφές
    - Κρύβουν την υλοποίηση
  - Επαναχρησιμοποίηση Κώδικα
    - Σύνθεση (συνάθροιση) - Composition (aggregation)
      - Αντικείμενα κλάσης μπορούν να περιληφθούν ως μέλη άλλων κλάσεων
    - Κληρονομικότητα
      - Νέες κλάσεις προκύπτουν από παλαιές

6.6 Εμβέλεια κλάσης και προσπέλαση μελών

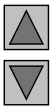
- Εμβέλεια κλάσης
  - Μέλη δεδομένα και συναρτήσεις
  - Εντός εμβέλειας
    - Μέλη κλάσης
      - Άμεσα προσπελάσιμα από όλα τα μέλη συναρτήσεις
      - Αναφορά με το όνομα
  - Εκτός εμβέλειας κλάσης
    - Αναφορά με handles
      - Όνομα αντικειμένου, αναφορά στο αντικείμενο, δείκτης στο αντικείμενο

6.6 Εμβέλεια κλάσης και προσπέλαση μελών

- Εμβέλεια συνάρτησης
  - Μεταβλητές δηλώνονται σε συναρτήσεις μέλη
  - Είναι γνωστές μόνο στη συνάρτηση
  - Μεταβλητές με ίδιο όνομα με μεταβλητές εμβέλειας κλάσης
    - Η εμβέλεια της μεταβλητής κλάσης «κρύβεται»
      - Προσπέλαση με τελεστή καθορισμού εμβέλειας (`::`)
        `ClassName::classVariableName`
  - Οι μεταβλητές είναι γνωστές στις συναρτήσεις που ορίζονται
  - Οι μεταβλητές καταστρέφονται μετά την ολοκλήρωση της συνάρτησης

6.6 Εμβέλεια κλάσης και προσπέλαση μελών

- Τελεστές για προσπέλαση μελών κλάσης
  - Ίδια με των `structs`
  - Επιλογή με τελεία (`.`)
    - Αντικείμενο
    - Αναφορά σε αντικείμενο
  - Επιλογή με βέλος (`->`)
    - Δείκτες
  - `timePtr->hour` είναι όμοιο με `( *timePtr ).hour`
    - Οι παρενθέσεις απαιτούνται
      - `*` έχει μικρότερη προτεραιότητα έναντι του `.`

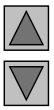


Outline

fig06_04.cpp (1 of 2)

```
1  // Fig. 6.4: fig06_04.cpp
2  // Demonstrating the class member access operators . and ->
3  //
4  // CAUTION: IN FUTURE EXAMPLES WE AVOID PUBLIC DATA!
5  #include <iostream>
6
7  using std::cout;
8  using std::endl;
9
10 // class Count definition
11 class Count {
12
13 public:
14    int x;
15
16    void print()
17    {
18       cout << x << endl;
19    }
20
21 }; // end class Count
22
```

Το **x public** χρησιμοποιείται ως παράδειγμα μόνο. Τυπικά τα δεδομένα είναι **private**.



Outline

fig06_04.cpp (2 of 2)

```
23 int main()
24 {
25    Count counter;               // create counter object
26    Count *counterPtr = &counter; // create pointer to counter
27    Count &counterRef = counter;  //
28
29    cout << "Assign 1 to x and print
30    counter.x = 1;          // assign 1
31    counter.print();        // call membe
32
33    cout << "Assign 2 to x and print u
34    counterRef.x = 2;       // assign 2 t
35    counterRef.print();     // call membe
36
37    cout << "Assign 3 to x and print us
38    counterPtr->x = 3;      // assign 3 to data member x
39    counterPtr->print(); // call member function print
40
41    return 0;
42
43 } // end main
```

Χρήση τελείας για την επιλογή μέλους δεδομένου από το αντικείμενο **counter**

Χρήση τελείας για την επιλογή μέλους στην αναφορά **counterRef** στο

Χρήση βέλους για το δείκτη **counterPtr** στο αντικείμενο.

fig06_04.cpp output (1 of 1)

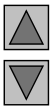
```
Assign 1 to x and print using the object's name: 1
Assign 2 to x and print using a reference: 2
Assign 3 to x and print using a pointer: 3
```

6.7 Διαχωρίζοντας τη διεπαφή από την υλοποίηση

- Διαχωρίζοντας τη διεπαφή από την υλοποίηση
  - Πλεονέκτημα
    - Διευκολύνει την τροποποίηση προγραμμάτων
  - Μειονέκτημα
    - Αρχεία Header
      - Μέρη υλοποίησης
        - Inline συναρτήσεις
      - Ιδέες από την υλοποίηση

6.7 Διαχωρίζοντας τη διεπαφή από την υλοποίηση

- Header αρχεία
  - Ορισμοί κλάσης και πρωτότυπα συναρτήσεων
  - Περιλαμβάνονται σε κάθε αρχείο
    - `#include`
  - Επέκταση ονόματος αρχείου `.h`
- Αρχεία κώδικα
  - Ορισμοί μελών συναρτήσεων
  - Ίδιο όνομα βάσης
    - Σύμβαση



Outline

time1.h (1 of 1)

```
1  // Fig. 6.5: time1.h
2  // Declaration of class Time.
3  // Member functions are defined in t
4
5  // prevent multiple inclusions of header file
6  #ifndef TIME1_H
7  #define TIME1_H
8
9  // Time abstract d
10 class Time {
11
12 public:
13    Time();
14    void setTime( int, int, int ); // set hour, minute, second
15    void printUniversal();         // print universal-time format
16    void printStandard();          // print standard-time format
17
18 private:
19    int hour;     // 0 - 23 (24-hour clock format)
20    int minute;   // 0 - 59
21    int second;   // 0 - 59
22
23 }; // end class Time
24
25 #endif
```

Κώδικας που προλαμβάνει πολλαπλές συμπεριλήψεις

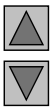
Κώδικας δεν περιλαμβάνεται

"If not de

sor

H.

Ονοματολογία:
το όνομα του αρχείου header με το underscore να αντικαθιστά την τελεία.



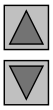
Outline

time1.cpp (1 of 3)

```
1  // Fig. 6.6: time1.cpp
2  // Member-function definitions for class Time.
3  #include <iostream>
4
5  using std::cout;
6
7  #include <iomanip>
8
9  using std::setfill;
10 using std::setw;
11
12 // include definition of class Time from time1.h
13 #include "time1.h"
14
15 // Time constructor initializes each data member to zero.
16 // Ensures all Time objects
17 Time::Time()
18 {
19    hour = minute = second =
20
21 } // end Time constructor
22
```

Περίληψη του αρχείου header **time1.h**.

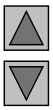
Το όνομα αρχείου μέσα σε quotes. Όταν χρησιμοποιούνται οι ανισότητες θεωρείται ότι είναι μέρος της C++ Standard Library.



Outline

time1.cpp (2 of 3)

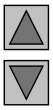
```
23 // Set new Time value using universal time. Perform validity
24 // checks on the data values. Set invalid values to zero.
25 void Time::setTime( int h, int m, int s )
26 {
27    hour = ( h >= 0 && h < 24 ) ? h : 0;
28    minute = ( m >= 0 && m < 60 ) ? m : 0;
29    second = ( s >= 0 && s < 60 ) ? s : 0;
30
31 } // end function setTime
32
33 // print Time in universal format
34 void Time::printUniversal()
35 {
36    cout << setfill( '0' ) << setw( 2 ) << hour << ":"
37         << setw( 2 ) << minute << ":"
38         << setw( 2 ) << second;
39
40 } // end function printUniversal
41
```

Outline

time1.cpp (3 of 3)

```
42 // print Time in standard format
43 void Time::printStandard()
44 {
45    cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 )
46         << ":" << setfill( '0' ) << setw( 2 ) << minute
47         << ":" << setw( 2 ) << second
48         << ( hour < 12 ? " AM" : " PM" );
49
50 } // end function printStandard
```

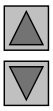


Outline

fig06_07.cpp (1 of 2)

```
1  // Fig. 6.7: fig06_07.cpp
2  // Program to test class Time.
3  // NOTE: This file must be compiled with time1.cpp.
4  #include <iostream>
5
6  using std::cout;
7  using std::endl;
8
9  // include definition of class Time
10 #include "time1.h"
11
12 int main()
13 {
14    Time t;  // instantiate object t of class Time
15
16    // output Time object t's initial values
17    cout << "The initial universal time is ";
18    t.printUniversal();   // 00:00:00
19    cout << "\nThe initial standard time is ";
20    t.printStandard();    // 12:00:00 AM
21
22    t.setTime( 13, 27, 6 );   // change time
23
```

Συμπερίληψη του **time1.h** για τη σωστή δημιουργία/ χειρισμό και καθορισμό του μεγέθους του αντικειμένου κλάσης **Time**.



Outline

fig06_07.cpp (2 of 2)

```
24    // output Time object t's new values
25    cout << "\n\nUniversal time after setTime is ";
26    t.printUniversal();   // 13:27:06
27    cout << "\nStandard time after setTime is ";
28    t.printStandard();    // 1:27:06 PM
29
30    t.setTime( 99, 99, 99 );  // attempt invalid settings
31
32    // output t's values after specifying invalid values
33    cout << "\n\nAfter attempting invalid settings:"
34         << "\nUniversal time: ";
35    t.printUniversal();   // 00:00:00
36    cout << "\nStandard time: ";
37    t.printStandard();    // 12:00:00 AM
38    cout << endl;
39
40    return 0;
41
42 } // end main
```

fig06_07.cpp output (1 of 1)

```
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
```

6.8 Έλεγχος πρόσβασης στα μέλη

- Επίπεδα πρόσβασης
  - **`private`**
    - Default
    - Προσβάσιμο σε μέλη συναρτήσεις και friends
  - **`public`**
    - Προσβάσιμο σε κάθε συνάρτηση που χειρίζεται αντικείμενο της κλάσης
  - **`protected`**
    - Προσβάσιμο στις υποκλάσεις
- Δεν υπάρχουν προσδιοριστικά για κλάσεις!
  - Δηλώνοντας μια κλάση σε έναν header την καθιστά public
  - Δεν υπάρχει η έννοια του πακέτου

Outline

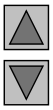
fig06_08.cpp (1 of 1)

```
1  // Fig. 6.8: fig06_08.cpp
2  // Demonstrate errors resulting from attempts
3  // to access private class members.
4  #include <iostream>
5
6  using std::cout;
7
8  // include definition of class Time from time1.h
9  #include "time1.h"
10
11 int main()
12 {
13    Time t;  // create Time obje
14
15
16    t.hour = 7;  // error: 'Time::hour' is not access
17
18    // error: 'Time::minute' is not accessible
19    cout << "minute = " << t.minute;
20
21    return 0;
22
23 } // end main
```

Το **`hour`** είναι **`private`**. Άρα η προσπάθεια προσπέλασης του δίνει λάθος.

Το **`minute`** είναι **`private`**. Άρα η προσπάθεια προσπέλασης του δίνει λάθος.

```
D:\cpphtp4_examples\ch06\Fig6_06\Fig06_06.cpp(16) : error C2248:
    'hour' : cannot access private member declared in class 'Time'
D:\cpphtp4_examples\ch06\Fig6_06\Fig06_06.cpp(19) : error C2248:
    'minute' : cannot access private member declared in class 'Time'
```



Outline

fig06_08.cpp output (1 of 1)

Τα λάθη που δημιουργούνται κατά την προσπέλαση μελών **`private`**.

6.8 Έλεγχος πρόσβασης στα μέλη

- Πρόσβαση σε μέλη κλάσης
  - Default **`private`**
  - Μπορούν να τεθούν ως **`private`**, **`public`**, **`protected`**
- **`struct`**
  - Default **`public`**
  - Μπορούν να τεθούν σε **`private`**, **`public`**, **`protected`**
- Πρόσβαση σε **`private`** δεδομένα κλάσης
  - Ελέγχονται με συναρτήσεις πρόσβασης (accessor methods)
    - Συνάρτηση Get
      - Ανάγνωση **`private`** δεδομένου
    - Συνάρτηση Set
      - Τροποποίηση **`private`** δεδομένου

6.9 Συναρτήσεις πρόσβασης και βοηθητικές συναρτήσεις

- Συναρτήσεις πρόσβασης
  - `public`
  - Διαβάζουν/ παρουσιάζουν δεδομένα
  - Δηλωτικές συναρτήσεις
    - Έλεγχος
- Βοηθητικές συναρτήσεις Utility functions (helper functions)
  - `private`
  - Υποστηρίζουν τη λειτουργία των `public` συναρτήσεων
  - Δεν έχουν δημιουργηθεί για χρήση από τον τελικό χρήστη

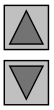
Outline

salesp.h (1 of 1)

```
1  // Fig. 6.9: salesp.h
2  // SalesPerson class definition.
3  // Member functions defined in salesp.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
13    void printAnnualSales();        // summarize
14
15 private:
16    double totalAnnualSales();      // utility function
17    double sales[ 12 ];             // 12 monthly sales figures
18
19 }; // end class SalesPerson
20
21 #endif
```

Η συνάρτηση Set κάνει ελέγχους.

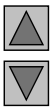
private βοηθητική συνάρτηση



Outline

salesp.cpp (1 of 3)

```
1  // Fig. 6.10: salesp.cpp
2  // Member functions for class SalesPerson.
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8  using std::fixed;
9
10 #include <iomanip>
11
12 using std::setprecision;
13
14 // include SalesPerson class definition from salesp.h
15 #include "salesp.h"
16
17 // initialize elements of array sales to 0.0
18 SalesPerson::SalesPerson()
19 {
20    for ( int i = 0; i < 12; i++ )
21       sales[ i ] = 0.0;
22
23 } // end SalesPerson constructor
24
```

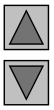



Outline

salesp.cpp (2 of 3)

```
25 // get 12 sales figures from the user at the keyboard
26 void SalesPerson::getSalesFromUser()
27 {
28    double salesFigure;
29
30    for ( int i = 1; i <= 12; i++ ) {
31       cout << "Enter sales amount for month " << i << ": ";
32       cin >> salesFigure;
33       setSales( i, salesFigure );
34
35    } // end for
36
37 } // end function getSalesFromUser
38
39 // set one of the 12 monthly sales figures; function subtracts
40 // one from month value for proper subscri
41 void SalesPerson::setSales( int month, doub
42 {
43    // test for valid month and amount values
44    if ( month >= 1 && month <= 12 && amount > 0 )
45       sales[ month - 1 ] = amount; // adjust for subscripts 0-11
46
47    else // invalid month or amount value
48       cout << "Invalid month or sales figure" << endl;
```

Η συνάρτηση Set κάνει ελέγχους.

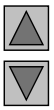


Outline

salesp.cpp (3 of 3)

```
49
50 } // end function setSales
51
52 // print total annual sales (with help of utility function)
53 void SalesPerson::printAnnualSales()
54 {
55    cout << setprecision( 2 ) << fixed
56         << "\nThe total annual sales are: $"
57         << totalAnnualSales() << endl; // call utility function
58
59 } // end function printAnnualSales
60
61 // private utility function to total annual sales
62 double SalesPerson::totalAnnualSales()
63 {
64    double total = 0.0;              // initialize total
65
66    for ( int i = 0; i < 12; i++ )  // summarize sales results
67       total += sales[ i ];
68
69    return total;
70
71 } // end function totalAnnualSales
```

private βοηθητική συνάρτηση της **printAnnualSales**, που πραγματοποιεί διαχείριση του **sales** array.



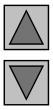
Outline

fig06_11.cpp (1 of 1)

```
1  // Fig. 6.11: fig06_11.cpp
2  // Demonstrating a utility function.
3  // Compile this program with salesp.cpp
4
5  // include SalesPerson class definition from salesp.h
6  #include "salesp.h"
7
8  int main()
9  {
10    SalesPerson s;          // create SalesPerson object
11
12    s.getSalesFromUser();  // note simple sequential coc
13    s.printAnnualSales();  // control structures in main
14
15    return 0;
16
17 } // end main
```

Σειριακή κλήση των συναρτήσεων. Η λογική είναι εμφωλευμένη σε συναρτήσεις μέλη.

```
Enter sales amount for month 1: 5314.76
Enter sales amount for month 2: 4292.38
Enter sales amount for month 3: 4589.83
Enter sales amount for month 4: 5534.03
Enter sales amount for month 5: 4376.34
Enter sales amount for month 6: 5698.45
Enter sales amount for month 7: 4439.22
Enter sales amount for month 8: 5893.57
Enter sales amount for month 9: 4909.67
Enter sales amount for month 10: 5123.45
Enter sales amount for month 11: 4024.97
Enter sales amount for month 12: 5923.92

The total annual sales are: $60120.59
```



Outline

fig06_11.cpp output (1 of 1)

6.10 Αρχικοποίηση αντικειμένων κλάσης: Constructors

- Constructors
  - Αρχικοποιεί μέλη δεδομένα
  - Ίδιο όνομα με την κλάση
  - Δεν έχει τύπο επιστροφής
- Initializers
  - Περνούν ως παράμετροι στον constructor

```
Class-type ObjectName(
  value1,value2,…);
```

6.11 Χρήση Default παραμέτρων στους Constructors

- Constructors
  - Μπορούν να προσδιορίζουν default παραμέτρους εισόδου
  - Default constructors
    - Defaults για όλες τις παραμέτρους
      Ή
    - Μπορεί να κληθούν χωρίς παραμέτρους
    - Μόνο ένας για κάθε κλάση
  - Αν δεν υπάρχει:
    - Δημιουργεί ο compiler έναν default χωρίς παραμέτρους (implicit default constructor)

Copy Constructor

- Copy constructor
  - Είναι μια συνάρτηση μέλους που αρχικοποιεί ένα αντικείμενο της κλάσης με βάση ένα υπάρχον αντικείμενο της ίδιας κλάσης

```
ClassName (const ClassName &old_obj);
```

  - Ο compiler παρέχει έναν default
  - Memberwise copy — member by member copy
- Ο **copy constructor** καλείται όταν:
  - Ένα αντικείμενο περνιέται ως παράμετρος σε συνάρτηση
  - Ένα αντικείμενο επιστρέφεται από μία συνάρτηση
  - Default pass-by-value
    - ***Αντίγραφο*** του αντικείμενου που περνάει, επιστρέφεται
- Deep vs shallow copy

6.16 Default Ανάθεση

- Ανάθεση αντικειμένων

```
Test & operator = (const Test &t);
```

Τελεστής ανάθεσης (=) – **Copy Assignment Operator**

  - Μπορεί να αναθέσει σε ένα ήδη υπάρχον αντικείμενο ένα άλλο ίδιου τύπου
  - Επίσης παρέχεται default
- Ο copy constructor καλείται για αρχικοποίηση, ενώ ο assignment operator για ανάθεση *ήδη υπάρχοντος* αντικειμένου

```
t2 = t1; // calls assignment operator, same as
"t2.operator=(t1);"
Test t3 = t1; // calls copy constructor, same as
"Test t3(t1);"
```

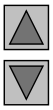
- Πρόβλημα όταν τα μέλη της κλάσης είναι δείκτες!

Outline

time2.h (1 of 1)

```
1  // Fig. 6.12: time2.h
2  // Declaration of class Time.
3  // Member functions defined in time2.cpp.
4
5  // prevent multiple inclusions of header file
6  #ifndef TIME2_H
7  #define TIME2_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    void setTime( int, int, int ); // set hour, minute, second
15    void printUniversal();         // print universal-time format
16    void printStandard();          // print standard-time format
17
18 private:
19    int hour;     // 0 - 23 (24-hour clock format)
20    int minute;   // 0 - 59
21    int second;   // 0 - 59
22
23 }; // end class Time
24
25 #endif
```

Default constructor προσδιορίζει όλες τις παραμέτρους

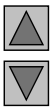


Outline

time2.cpp (1 of 3)

```
1  // Fig. 6.13: time2.cpp
2  // Member-function definitions for class Time.
3  #include <iostream>
4
5  using std::cout;
6
7  #include <iomanip>
8
9  using std::setfill;
10 using std::setw;
11
12 // include definition of class Time from time2.h
13 #include "time2.h"
14
15 // Time constructor initializes each data member to zero;
16 // ensures all Time objects start in a consistent state
17 Time::Time( int hr, int min, int sec )
18 {
19    setTime( hr, min, sec );  // validate and set time
20
21 } // end Time constructor
22
```

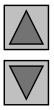
O Constructor καλεί τη **`setTime`** για να δηλώσει ή να ελέγξει τιμές.



Outline

time2.cpp (2 of 3)

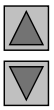
```
23 // set new Time value using universal time, perform validity
24 // checks on the data values and set invalid values to zero
25 void Time::setTime( int h, int m, int s )
26 {
27    hour = ( h >= 0 && h < 24 ) ? h : 0;
28    minute = ( m >= 0 && m < 60 ) ? m : 0;
29    second = ( s >= 0 && s < 60 ) ? s : 0;
30
31 } // end function setTime
32
33 // print Time in universal format
34 void Time::printUniversal()
35 {
36    cout << setfill( '0' ) << setw( 2 ) << hour << ":"
37         << setw( 2 ) << minute << ":"
38         << setw( 2 ) << second;
39
40 } // end function printUniversal
41
```



Outline

time2.cpp (3 of 3)

```
42 // print Time in standard format
43 void Time::printStandard()
44 {
45    cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 )
46         << ":" << setfill( '0' ) << setw( 2 ) << minute
47         << ":" << setw( 2 ) << second
48         << ( hour < 12 ? " AM" : " PM" );
49
50 } // end function printStandard
```

Outline

fig06_14.cpp (1 of 2)

```
1  // Fig. 6.14: fig06_14.cpp
2  // Demonstrating a default constructor for class Time.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  // include definition of class Time from time2.h
9  #include "time2.h"
10
11 int main()
12 {
13    Time t1;               // all arguments defaulted
14    Time t2( 2 );          // minute and second defaulted
15    Time t3( 21, 34 );     // second defaulted
16    Time t4( 12, 25, 42 ); // all values specified
17    Time t5( 27, 74, 99 ); // all bad values specified
18
19    cout << "Constructed with:\n\n"
20         << "all default arguments:\n  ";
21    t1.printUniversal();  // 00:00:00
22    cout << "\n  ";
23    t1.printStandard();   // 12:00:00 AM
24
```

Αρχικοποίηση του αντικειμένου **Time** με default παραμέτρους.

Initialize **Time** object with invalid values; validity checking will set values to **0**.

```
25    cout << "\n\nhour specified; default minute and second:\n  ";
26    t2.printUniversal();  // 02:00:00
27    cout << "\n  ";
28    t2.printStandard();   // 2:00:00 AM
29
30    cout << "\n\nhour and minute specified; default second:\n  ";
31    t3.printUniversal();  // 21:34:00
32    cout << "\n  ";
33    t3.printStandard();   // 9:34:00 PM
34
35    cout << "\n\nhour, minute, and second specified:\n  ";
36    t4.printUniversal();  // 12:25:42
37    cout << "\n  ";
38    t4.printStandard();   // 12:25:42 PM
39
40    cout << "\n\nall invalid values specified:\n  ";
41    t5.printUniversal();  // 00:00:00
42    cout << "\n  ";
43    t5.printStandard();   // 12:00:00 AM
44    cout << endl;
45
46    return 0;
47
48 } // end main
```

t5 με λάθος παραμέτρους, οπότε οι τιμές γίνονται όλες μηδέν (**0**).

Outline

fig06_14.cpp (2 of 2)


```
Constructed with:

all default arguments:
  00:00:00
  12:00:00 AM

hour specified; default minute and second:
  02:00:00
  2:00:00 AM

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

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

all invalid values specified:
  00:00:00
  12:00:00 AM
```

Outline

fig06_14.cpp output (1 of 1)

6.12 Destructors

- Destructors
  - Ειδικές συναρτήσεις μέλη
  - Όμοιο όνομα με την κλάση
    - Προηγείται η άρνηση (~)
  - Χωρίς παραμέτρους
  - Δεν επιστρέφει
  - Δεν υπερφορτώνεται
  - Κάνει απελευθέρωση μνήμης
  - Αν δεν υπάρχει destructor
    - Ο Compiler δημιουργεί έναν άδειο

6.13 Καλώντας Constructors και Destructors

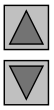
- Constructors και destructors
  - Καλούνται από τον compiler
- Σειρά κλήσης των συναρτήσεων
  - Εξαρτάται από τη σειρά εκτέλεσης
    - Όταν η εκτέλεση μπαίνει ή φεύγει από την εμβέλεια των αντικειμένων
  - Γενικά, ο destructor καλεί σε αντίστροφη σειρά από τους constructor

6.13 Καλώντας Constructors και Destructors

- Σειρά της κλήσης constructor, destructor
  - Αντικείμενα Global εμβέλειας
    - Constructors
      - Πριν από κάθε άλλη συνάρτηση (και της `main`)
    - Destructors
      - Όταν τερματίζει η `main` (ή καλείται η `exit`)
      - Δε καλείται με τερματισμό `abort`
  - Τοπικά αντικείμενα
    - Constructors
      - Όταν ορίζονται
        - Όταν η εκτέλεση γίνει εντός εμβέλειας
    - Destructors
      - Όταν τα αντικείμενα είναι εκτός εμβέλειας
        - Η εκτέλεση φεύγει από το block όπου ορίζονται
      - Δε καλείται σε τερματισμό `exit` ή `abort`

6.13 Καλώντας Constructors και Destructors

- Σειρά της κλήσης constructor, destructor
  - `static` τοπικά αντικείμενα
    - Constructors
      - Ακριβώς μία φορά
      - Όταν η εκτέλεση φτάσει στον ορισμό του αντικειμένου
    - Destructors
      - Όταν η `main` τερματίσει ή κληθεί `exit`
      - Δε καλείται στο `abort`



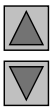
Outline

create.h (1 of 1)

```
1  // Fig. 6.15: create.h
2  // Definition of class CreateAndDestroy.
3  // Member functions defined in create.cpp.
4  #ifndef CREATE_H
5  #define CREATE_H
6
7  class CreateAndDestroy {
8
9  public:
10    CreateAndDestroy( int, char * );  // constructor
11    ~CreateAndDestroy();
12
13 private:
14    int objectID;
15    char *message;
16
17 }; // end class CreateAndDestroy
18
19 #endif
```

Constructor και destructor συναρτήσεις.

private members to show order of constructor, destructor function calls.

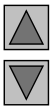


Outline

create.cpp (1 of 2)

```
1  // Fig. 6.16: create.cpp
2  // Member-function definitions for class CreateAndDestroy
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  // include CreateAndDestroy class definition from create.h
9  #include "create.h"
10
11 // constructor
12 CreateAndDestroy::CreateAndDestroy(
13    int objectNumber, char *messagePtr )
14 {
15    objectID = objectNumber;
16    message = messagePtr;
17
18    cout << "Object " << objectID << "   constructor runs   "
19         << message << endl;
20
21 } // end CreateAndDestroy constructor
22
```

Εκτύπωση μηνύματος που δείχνει την κλήση του constructor.

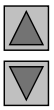


Outline

create.cpp (2 of 2)

```
23 // destructor
24 CreateAndDestroy::~CreateAndDestroy()
25 {
26    // the following line is for pedag
27    cout << ( objectID == 1 || objectI
28
29    cout << "Object " << objectID << "    destructor runs    "
30         << message << endl;
31
32 } // end ~CreateAndDestroy destructor
```

Εκτύπωση μηνύματος που δείχνει την κλήση του destructor.



Outline

fig06_17.cpp (1 of 3)

```
1  // Fig. 6.17: fig06_17.cpp
2  // Demonstrating the order in which constructors and
3  // destructors are called.
4  #include <iostream>
5
6  using std::cout;
7  using std::endl;
8
9  // include CreateAndDestroy class definition from create.h
10 #include "create.h"
11
12 void create( void );   // prototype
13
14 // global object
15 CreateAndDestroy first( 1, "(global before main)" );
16
17 int main()
18 {
19    cout << "\nMAIN FUNCTION: EXECUTION
20
21    CreateAndDestroy second( 2, "(local automatic in main)" );
22
23    static CreateAndDestroy third(
24       3, "(local static in main)" );
25
```

Δημιουργία μεταβλητής με εμβέλεια global.

automatic αντικείμενο.

static αντικείμενο.

```
26    create();  // call function to create objects
27
28    cout << "\nMAIN FUNCTION: EXECUTION RESUMES" << endl;
29
30    CreateAndDestroy fourth(
31
32    cout << "\nMAIN FUNCTION: EXECUTION ENDS" << endl;
33
34    return 0;
35
36 } // end main
37
38 // function to create object
39 void create( void )
40 {
41    cout << "\nCREATE FUNCTIO
42
43    CreateAndDestroy fifth( 5
44
45    static CreateAndDestroy s
46       6, "(local static in c
47
48    CreateAndDestroy seventh(
49       7, "(local automatic in create)" );
50
```

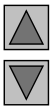
Δημιουργία των automatic αντικειμένων.

automatic αντικείμενο.

automatic αντικείμενο σε συνάρτηση.

static αντικείμενο σε συνάρτηση.

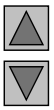
automatic αντικείμενο σε συνάρτηση.



Outline

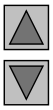
fig06_17.cpp
(2 of 3)


```
51    cout << "\nCREATE FUNCTION: EXECUTION ENDS\" << endl;
52
53 } // end function create
```



Outline

fig06_17.cpp
(3 of 3)



Outline

fig06_17.cpp output (1 of 1)

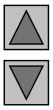
```
Object 1   constructor runs   (global before main)

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
Object 4   destructor runs    (local automatic in main)
Object 2   destructor runs    (local automatic in main)
Object 6   destructor runs    (local static in create)
Object 3   destructor runs    (local static in main)

Object 1   destructor runs    (global before main)
```

Τα **static** αντικείμενα

Τα **static** αντικείμενα δημιουργούνται με την κλήση της συνάρτησης και καταστρέφονται με τον τερματισμό της **main**.

6.14 Χρήση των Set και Get Συναρτήσεων

- Συνάρτηση Set
  - Πραγματοποιεί ελέγχους εγκυρότητας προτού γίνουν μετατροπές σε δεδομένα **private**
  - Ειδοποιεί αν υπάρχουν λανθασμένες τιμές
- Συνάρτηση Get
  - Ελέγχει τη μορφή των δεδομένων που επιστρέφονται



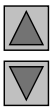
Outline

time3.h (1 of 2)

```
1  // Fig. 6.18: time3.h
2  // Declaration of class Time.
3  // Member functions defined in time3.cpp
4
5  // prevent multiple inclusions of header file
6  #ifndef TIME3_H
7  #define TIME3_H
8
9  class Time {
10
11 public:
12    Time( int = 0, int = 0, int = 0 );  // default constructor
13
14    // set functions
15    void setTime( int, int, int );  // set hour, minute, second
16    void setHour( int );    // set hour
17    void setMinute( int ); // set minute
18    void setSecond( int ); // set second
19
20    // get functions
21    int getHour();          // return hour
22    int getMinute();        // return minute
23    int getSecond();        // return second
24
```

Set.

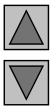
Get.



Outline

time3.h (2 of 2)

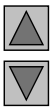
```
25    void printUniversal(); // output universal-time format
26    void printStandard();  // output standard-time format
27
28 private:
29    int hour;              // 0 - 23 (24-hour clock format)
30    int minute;            // 0 - 59
31    int second;            // 0 - 59
32
33 }; // end clas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



Outline

time3.cpp (1 of 4)

```
1  // Fig. 6.19: time3.cpp
2  // Member-function definitions for Time class.
3  #include <iostream>
4
5  using std::cout;
6
7  #include <iomanip>
8
9  using std::setfill;
10 using std::setw;
11
12 // include definition of class Time from time3.h
13 #include "time3.h"
14
15 // constructor function to initialize private data;
16 // calls member function setTime to set variables;
17 // default values are 0 (see class definition)
18 Time::Time( int hr, int min, int sec )
19 {
20    setTime( hr, min, sec );
21
22 } // end Time constructor
23
```

Outline

time3.cpp (2 of 4)

```
24 // set hour, minute and second values
25 void Time::setTime( int h, int m, int s )
26 {
27    setHour( h );
28    setMinute( m );
29    setSecond( s );
30
31 } // end function setTime
32
33 // set hour value
34 void Time::setHour( int h )
35 {
36    hour = ( h >= 0 && h < 24 ) ? h : 0;
37
38 } // end function setHour
39
40 // set minute value
41 void Time::setMinute( int m )
42 {
43    minute = ( m >= 0 && m < 60 ) ? m : 0;
44
45 } // end function setMinute
46
```

Κλήση set συνάρτησης για έλεγχο ορθότητας των δεδομένων

Οι συναρτήσεις Set ελέγχουν τα δεδομένα προτού τα μετατρέψουν.

Outline

time3.cpp (3 of 4)

```
47 // set second value
48 void Time::setSecond( int s )
49 {
50    second = ( s >= 0 && s < 60 ) ? s : 0;
51
52 } // end function setSecond
53
54 // return hour value
55 int Time::getHour()
56 {
57    return hour;
58
59 } // end function getHour
60
61 // return minute value
62 int Time::getMinute()
63 {
64    return minute;
65
66 } // end function getMinute
67
```

Οι συναρτήσεις Set ελέγχουν τα δεδομένα προτού τα μετατρέψουν.

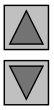
Οι συναρτήσεις Get επιτρέπουν την ανάγνωση των δεδομένων.

Outline

time3.cpp (4 of 4)

```
68 // return second value
69 int Time::getSecond()
70 {
71    return second;
72
73 } // end function getSecond
74
75 // print Time in universal format
76 void Time::printUniversal()
77 {
78    cout << setfill( '0' ) << setw( 2 ) << hour << ":"
79         << setw( 2 ) << minute << ":"
80         << setw( 2 ) << second;
81
82 } // end function printUniversal
83
84 // print Time in standard format
85 void Time::printStandard()
86 {
87    cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 )
88         << ":" << setfill( '0' ) << setw( 2 ) << minute
89         << ":" << setw( 2 ) << second
90         << ( hour < 12 ? " AM" : " PM" );
91
92 } // end function printStandard
```

Οι συναρτήσεις Get επιτρέπουν την ανάγνωση των δεδομένων.

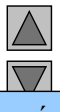


Outline

fig06_20.cpp (1 of 3)

```
1  // Fig. 6.20: fig06_20.cpp
2  // Demonstrating the Time class set and get functions
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  // include definition of class Time from time3.h
9  #include "time3.h"
10
11 void incrementMinutes( Time &, const int );  // prototype
12
13 int main()
14 {
15    Time t;              // create Time object
16
17    // set time using individual set functions
18    t.setHour( 17 );     // set hour to valid value
19    t.setMinute( 34 );   // set minute to valid value
20    t.setSecond( 25 );   // set second to valid value
21
```

Κλήση set για να οριστούν οι έγκυρες τιμές.



Outline

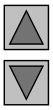
(2 of 3)

```
22    // use get functions to obtain hour, minute and second
23    cout << "Result of setting all valid values:\n"
24         << "  Hour: " << t.getHour()
25         << "  Minute: " << t.getMinute()
26         << "  Second: " << t.getSecond();
27
28    // set time using individual set functions
29    t.setHour( 234 );    // invalid hour set to 0
30    t.setMinute( 43 );   // set minute to valid value
31    t.setSecond( 6373 ); // invalid second set to 0
32
33    // display hour, minute and second after setting
34    // invalid hour and second values
35    cout << "\n\nResult of attempting to set invalid hour and"
36         << " second:\n  Hour: " << t.getHour()
37         << "  Minute: " << t.getMinute()
38         << "  Second: " << t.getSecond() << "\n\n";
39
40    t.setTime( 11, 58, 0 );    // set time
41    incrementMinutes( t, 3 );  // increment t's minute by 3
42
43    return 0;
44
45 } // end main
46
```

Προσπάθεια ορισμού λανθασμένων τιμών με χρήση set.

Οι λάθος τιμές έχουν ως αποτέλεσμα να οριστούν οι μεταβλητές σε **0**.

Μετατροπή δεδομένων με τη **setTime**.

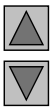


Outline

fig06_20.cpp

```
47 // add specified number of minutes to a Time object
48 void incrementMinutes( Time &tt, const int count )
49 {
50    cout << "Incrementing minute " << count
51         << " times:\nStart time: ";
52    tt.printStandard();
53
54    for ( int i = 0; i < count; i++ ) {
55       tt.setMinute( ( tt.getMinute() + 1 ) % 60 );
56
57       if ( tt.getMinute() == 0 )
58          tt.setHour( ( tt.getHour() + 1 ) % 24);
59
60       cout << "\nminute + 1: ";
61       tt.printStandard();
62
63    } // end for
64
65    cout << endl;
66
67 } // end function incrementMinutes
```

Με τις get διαβάζονται δεδομένα και με τις set τροποποιούνται.



Outline

fig06_20.cpp output (1 of 1)

```
Result of setting all valid values:
  Hour: 17  Minute: 34  Second: 25

Result of attempting to set invalid hour and second:
  Hour: 0  Minute: 43  Second: 0

Incrementing minute 3 times:
Start time: 11:58:00 AM
minute + 1: 11:59:00 AM
minute + 1: 12:00:00 PM
minute + 1: 12:01:00 PM
```

Προσπάθεια να οριστούν λάθος τιμές που καταλήγει σε μήνυμα λάθους και τιμή μηδέν (0)

6.15 Προσοχή: Επιστροφή αναφοράς σε δεδομένα private

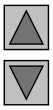
- Αναφορά σε αντικείμενο
  - Alias for name of object
  - Lvalue
    - Μπορεί να λάβει τιμή σε ανάθεση
      - Αλλάζει το αρχικό αντικείμενο
- Επιστροφή αναφοράς
  - `public` συναρτήσεις μπορούν να επιστρέφουν μη-`const` αναφορές σε δεδομένα `private`
    - Κατά συνέπεια ο χρήστης μπορεί να τροποποιεί δεδομένα `private`

Outline

time4.h (1 of 1)

```
1  // Fig. 6.21: time4.h
2  // Declaration of class Time.
3  // Member functions defined in time4.cpp
4
5  // prevent multiple inclusions of header file
6  #ifndef TIME4_H
7  #define TIME4_H
8
9  class Time {
10
11 public:
12    Time( int = 0, int = 0, int = 0 );
13    void setTime( int, int, int );
14    int getHour();
15
16    int &badSetHour( int );  // DANGEROUS re
17
18 private:
19    int hour;
20    int minute;
21    int second;
22
23 }; // end class Time
24
25 #endif
```

Συνάρτηση που δείχνει τα αποτελέσματα επιστροφής αναφοράς σε δεδομένα **`private`**.

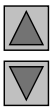


Outline

time4.cpp (1 of 2)

3

```
1  // Fig. 6.22: time4.cpp
2  // Member-function definitions for Time class.
3
4  // include definition of class Time from time4.h
5  #include "time4.h"
6
7  // constructor function to initialize private data;
8  // calls member function setTime to set variables;
9  // default values are 0 (see class definition)
10 Time::Time( int hr, int min, int sec )
11 {
12    setTime( hr, min, sec );
13
14 } // end Time constructor
15
16 // set values of hour, minute and second
17 void Time::setTime( int h, int m, int s )
18 {
19    hour = ( h >= 0 && h < 24 ) ? h : 0;
20    minute = ( m >= 0 && m < 60 ) ? m : 0;
21    second = ( s >= 0 && s < 60 ) ? s : 0;
22
23 } // end function setTime
24
```

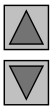



Outline

time4.cpp (2 of 2)

```
25 // return hour value
26 int Time::getHour()
27 {
28    return hour;
29
30 } // end function getHour
31
32 // POOR PROGRAMMING PRACTICE:
33 // Returning a reference to a private data member.
34 int &Time::badSetHour( int hh )
35 {
36    hour = ( hh >= 0 && hh < 24 )
37
38    return hour;  // DANGEROUS reference return
39
40 } // end function badSetHour
```

Επιστροφή αναφοράς σε **`private`** μεταβλητή **`hour`**.



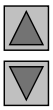
Outline

fig06_23.cpp (1 of 2)

```
1  // Fig. 6.23: fig06_23.cpp
2  // Demonstrating a public member function that
3  // returns a reference to a private data member.
4  #include <iostream>
5
6  using std::cout;
7  using std::endl;
8
9  // include definition of class Time from time4.h
10 #include "time4.h"
11
12 int main()
13 {
14    Time t;
15
16    // store in hourRef the reference returned by badSet
17    int &hourRef = t.badSetHour( 20 );
18
19    cout << "Hour before modificati
20
21    // use hourRef to set invalid v
22    hourRef = 30;
23
24    cout << "\nHour after modification: " << t.getHour();
25
```

Η **badSetHour** επιστρέφει αναφορά σε **private** μεταβλητή **hour**.

Η αναφορά επιτρέπει να οριστεί τιμή στη **private hour**.



Outline

fig06_23.cpp (2 of 2)

fig06_23.cpp output (1 of 1)

```
26    // Dangerous: Function call that returns
27    // a reference can be used as an lvalue!
28    t.badSetHour( 12 ) = 74;
29
30    cout << "\n\n*********************************\n"
31         << "POOR PROGRAMMING PRACTIC
32         << "badSetHour as an lvalue,
33         << t.getHour()
34         << "\n*********************************" << endl;
35
36    return 0;
37
38 } // end main
```

Μπορεί να χρησιμοποιηθεί κλήση συνάρτησης σε ανάθεση λάθους τιμής

```
Hour before modification: 20
Hour after modification: 30

*********************************
POOR PROGRAMMING PRACTICE!!!!!!!!
badSetHour as an lvalue, Hour: 74
*********************************
```

Η επιστροφή αναφοράς επιτρέπει τον ορισμό λάθους τιμής σε **private** δεδομένα.

6.17 Επαναχρησιμοποίηση Κώδικα

- Επαναχρησιμοποίηση Κώδικα
  - Βιβλιοθήκες κλάσεων
    - Καλώς ορισμένες
    - Προσεκτικά ελεγμένες
    - Καλώς τεκμηριωμένες
    - Μεταφέρσιμες - Portable
    - Διαθέσιμες
  - Επιταχύνουν την ανάπτυξη ποιοτικού λογισμικού
    - Rapid applications development (RAD)
  - Προβλήματα
    - Κατάλογοι - λίστες
    - Θέματα αδειών
    - Μηχανισμοί προστασίας

Differences classes structures

- Although classes and structures are similar in both the way they are declared and how they are used, there are some significant differences. Classes are reference types and structs value types. A structure is allocated on the stack when it is declared and the variable is bound to its address. It directly contains the value. Classes are different because the memory is allocated as objects on the heap. Variables are rather managed pointers on the stack which point to the objects. They are references.
- Classes are private by default (members inheritance) structs are public.
- A structure cannot be inherited by a class where as a class can be inherited by a structure.
- Structures require some more than classes. For example, you need to explicitly create a default constructor which takes no arguments to initialize the struct and its members. The compiler will create a default one for classes. All fields and properties of a struct must have been initialized before an instance is created. Structs are more suitable for smaller constructs of data.

Difference new(), malloc()

- When you new an object, space for the object is not only allocated but the object's constructor is called. And similarly when you delete an object, the object's destructor is called before the memory is released. If you use malloc and free, the destructor and constructor do not get called respectively and obviously, this simply won't do in C++ except in certain very rare situations where you have classes without any specific destructor/constructors.