



BİL102 Nesne Yönelimli Programlama

Dr. Öğr. Üyesi Yavuz CANBAY
Kahramanmaraş Sütçü İmam Üniversitesi
Bilgisayar Mühendisliği Bölümü



Konu 5

- › Friend fonksiyonlar ve sınıflar
- › This pointer (işaretçisi)

Friend Fonksiyonlar ve Friend Sınıflar

- › Friend olarak tanımlanan bir fonksiyon veya sınıf, başka bir sınıfın private ve protected üyelerine ulaşmaya imkan tanır
- › Dikkat edilmesi gereken husus: friend fonksiyonlar sınıfın üyesi değildirler.
- › Fonksiyonlarda olduğu gibi Sınıflarda da friend tanımlama işlemi mevcuttur.
- › A sınıfı B sınıfında friend olarak tanımlı ise, bu B sınıfını A sınıfında friend yapmaz.
- › Bir C sınıfında B sınıfı friend ise, bu A sınıfı için C sınıfını friend yapmaz.

Erişim	Sınıf İçerisinden Erişim	Türetilmiş Sınıftan Erişim	Dışarıdan Erişim
public	evet	evet	evet
protected	evet	evet	hayır
private	evet	hayır	hayır

Friend bildirimi

- › Friend bildirimi;
 - › Fonksiyon ismi ve türünden önce yazılır.
 - › ***friend*** int myFunction(int x);
 - › Sınıf ismi ve türünden önce yazılırlar.
 - › ***friend*** class Vehicle;

Örnek1-friend fonksiyon

```
1 // Fig. 7.5: fig07_05.cpp
2 // Friends can access private members of a class.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 // Modified Count class
9 class Count {
10     friend void setX( Count &, int ); // friend declaration
11 public:
12     Count() { x = 0; } // constructor
13     void print() const { cout << x << endl; } // output
14 private:
15     int x; // data member
16 };
17
18 // Can modify private data of Count because
19 // setX is declared as a friend function of Count
20 void setX( Count &c, int val )
```

setX count türü bir friend' dir (**private** dataya ulaşabilir).

setX fonksiyonu Count' un bir üyesi olmadığından normal olarak tanımlanabilir.

Örnek1-friend fonksiyon-devam

```
21 {  
22     c.x = val;  // legal: setX is a friend of Count  
23 }  
24  
25 int main()  
26 {  
27     Count counter;  
28  
29     cout << "counter.x after instantiation: ";  
30     counter.print();  
31     cout << "counter.x after call to setX friend function: ";  
  
32     setX( counter, 8 );  // set x with a friend  
  
33     counter.print();  
  
34     return 0;  
  
35 }
```

```
counter.x after instantiation: 0  
counter.x after call to setX friend function: 8
```

Örnek2-friend fonksiyon

```
1 // Fig. 7.6: fig07_06.cpp
2 // Non-friend/non-member functions cannot access
3 // private data of a class.
4 #include <iostream>
5
6 using std::cout;
7 using std::endl;
8
9 // Modified Count class
10 class Count {
11 public:
12     Count() { x = 0; } // constructor
13     void print() const { cout << x << endl; } // output
14 private:
15     int x; // data member
16 };
17
18 // Function tries to modify private data of Count.
19 // but cannot because it is not a friend of Count.
20 void cannotSetX( Count &c, int val )
21 {
22     c.x = val; // ERROR: 'Count::x' is not accessible
23 }
```

cannotSetX count için bir
friend olmadığından private
dataya ulaşamaz

Örnek2-friend fonksiyon-devam

```
24
25 int main()
26 {
27     Count counter;
28
29     cannotSetX( counter, 3 ); // cannotSetX is not a friend
30     return 0;
31 }
```

```
Compiling...
Fig07_06.cpp
D:\books\2000\cpphttp3\examples\Ch07\Fig07_06\Fig07_06.cpp(22) :
error C2248: 'x' : cannot access private member declared in
class 'Count'
D:\books\2000\cpphttp3\examples\Ch07\Fig07_06\
Fig07_06.cpp(15) : see declaration of 'x'
Error executing cl.exe.

test.exe - 1 error(s), 0 warning(s)
```

Private dataya ulaşamadığından
oluşan bir derleyici hatası

Hadi Uygulayalım 😊

› Friend tanımlı Fonksiyon

```
1  #include <iostream>
2  using namespace std;
3  class Box {
4      friend void printWidth( Box box );
5
6      private:
7          double width;
8
9      public:
10         void setWidth( double wid );
11     };
12     // Member function definition
13     void Box::setWidth( double wid ) {
14         width = wid;
15     }
16     // Note: printWidth() is not a member function of any class.
17     void printWidth( Box box ) {
18         /* Because printWidth() is a friend of Box, it can
19         directly access any member of this class */
20         cout << "Width of box : " << box.width << endl;
21     }
22     // Main function for the program
23     int main() {
24         Box box;
25         // set box width without member function
26         box.setWidth(10.0);
27         // Use friend function to print the width.
28         printWidth( box );
29         return 0;
30     }
```

Hadi Uygulayalım 😊

› Friend tanımlı Sınıf

```
1  #include <iostream>
2  using namespace std;
3  class A {
4  private:
5      int a;
6
7  public:
8      A() { a = 0; }
9      friend class B; // Friend Class
10 };
11
12 class B {
13 private:
14     int b;
15
16 public:
17     void showA(A& x)
18     {
19         // Since B is friend of A, it can access private members of A
20         cout << "A::a=" << x.a;
21     }
22 };
23
24 int main()
25 {
26     A a;
27     B b;
28     b.showA(a);
29     return 0;
30 }
```

This Pointerı (işaretçisi)

- › **This** pointeri her nesnenin kendi adresine ulaşımı sağlar.
- › Nesnenin bir parçası değildir, nesnenin üyelerine erişmek için kullanılacak bir pointer'dır.

This Pointeri (işaretçisi)

This pointer kullanım örnekleri;

this->x

veya

(*this).x

- › Fonksiyon kendi nesnesinin adresi ile geri döner ve bu nesnenin diğer fonksiyonlarda da kullanılmasına imkan sağlar:

```
{ return *this; }
```

Örnek1

```
1 // Fig. 7.7: fig07_07.cpp
2 // Using the this pointer to refer to object members.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 class Test {
9 public:
10     Test( int = 0 );           // default constructor
11     void print() const;
12 private:
13     int x;
14 };
15
16 Test::Test( int a ) { x = a; } // constructor
17
18 void Test::print() const       // ( ) around
19 {
20     cout << "        x = " << x
21         << "\n  this->x = " << this->x
22         << "\n(*this).x = " << ( *this ).x << endl;
23 }
24
```

x 'i -> operatörü
kullanarak ekrana
yazdırmak

Direkt olarak x
ekranda görünür.

Nokta operatörü(.) kullanılarak x
görüntülenir fakat (.) operatörünün
öncelik seviyesi * operatöründen
daha fazla olduğu için parantez
kullanılmalıdır

Örnek1-devam

```
25 int main()
26 {
27     Test testObject( 12 );
28
29     testObject.print();
30
31     return 0;
32 }
```

```
    x = 12
this->x = 12
(*this).x = 12
```

Her üç metot da aynı sonucu verir

This Pointer'ının Kullanımı

- › İç-içe fonksiyon üyesi çağrımını mümkün kılar.

Time t;

t.setHour(1).setMinute(2).setSecond(3);

- › Yukarıdaki bu komut çalışınca:
 - › **t.setHour(1)** önce çalıştırılır, geriye **this** (t'nin adresi) döner. Yani ifade aşağıdaki gibi olur:
 - › **t.setMinute(2).setSecond(3);**
 - › **t.setMinute(2)** kısmı çalışınca nesnenin adresini tekrar döndürerek **t.setSecond(3);** halini alır.
 - › **t.setSecond(3)**, nesnenin adresini tekrar döndürerek **t** olur.

Örnek2

```
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
```

Dikkat **Time &...** fonksiyonu
Time nesnesinin adresini
döndürür.

Örnek2-devam

```
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
33 #endif
```

Örnek2-devam

```
34 // Fig. 7.8: time.cpp
35 // Member function definitions for Time class.
36 #include <iostream>
37
38 using std::cout;
39
40 #include "time6.h"
41
42 // Constructor function to initialize private data.
43 // Calls member function setTime to set variables.
44 // Default values are 0 (see class definition).
45 Time::Time( int hr, int min, int sec )
46     { setTime( hr, min, sec ); }
47
```

Örnek2-devam

```
48 // Set the values of hour, minute, and second.
49 Time &Time::setTime( int h, int m, int s )
50 {
51     setHour( h );
52     setMinute( m );
53     setSecond( s );
54     return *this;    // enables cascading
55 }
56
57 // Set the hour value
58 Time &Time::setHour( int h )
59 {
60     hour = ( h >= 0 && h < 24 ) ? h : 0;
61
62     return *this;    // enables cascading
63 }
```

***this** değerinin döndürülmesi ile iç-içe fonksiyonlar çağırılabilir.

Örnek2-devam

```
65 // Set the minute value
66 Time &Time::setMinute( int m )
67 {
68     minute = ( m >= 0 && m < 60 ) ? m : 0;
69
70     return *this;    // enables cascading
71 }
72
73 // Set the second value
74 Time &Time::setSecond( int s )
75 {
76     second = ( s >= 0 && s < 60 ) ? s : 0;
77
78     return *this;    // enables cascading
79 }
80
81 // Get the hour value
82 int Time::getHour() const { return hour; }
83
```


Örnek2-devam

```
84 // Get the minute value
85 int Time::getMinute() const { return minute; }
86
87 // Get the second value
88 int Time::getSecond() const { return second; }
89
90 // Display military format time: HH:MM
91 void Time::printMilitary() const
92 {
93     cout << ( hour < 10 ? "0" : "" ) << hour << ":"
94         << ( minute < 10 ? "0" : "" ) << minute;
95 }
96
97 // Display standard format time: HH:MM:SS AM (or PM)
98 void Time::printStandard() const
99 {
100     cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 )
101         << ":" << ( minute < 10 ? "0" : "" ) << minute
102         << ":" << ( second < 10 ? "0" : "" ) << second
103         << ( hour < 12 ? " AM" : " PM" );
104 }
```

Örnek2-devam

```
105// Fig. 7.8: fig07_08.cpp
106// Cascading member function calls together
107// with the this pointer
108#include <iostream>
109
110using std::cout;
111using std::endl;
112
113#include "time6.h"
114
115int main()
116{
117    Time t;
118
119    t.setHour( 18 ).setMinute( 30 ).setSecond( 22 );
120    cout << "Military time: ";
121    t.printMilitary();
```

İç-içe fonksiyon çağırılmasına dikkat!



Örnek2-devam

```
122  cout << "\nStandard time: ";
123  t.printStandard();
124
125  cout << "\n\nNew standard time: ";
126  t.setTime( 20, 20, 20 ).printStandard();
127  cout << endl;
128
129  return 0;
130 }
```

printStandard fonksiyonu nesnenin adresini döndürmediği için sadece en sondaki iç-içe fonksiyon olabilir.

Yani **t.printStandard().setTime()** ; yazılsa idi compiler hata verirdi.

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

New standard time: 8:20:20 PM

Hadi Uygulayalım 😊

```
1  #include <iostream>
2
3  using namespace std;
4
5  class Box {
6  public:
7      // Constructor definition
8      Box(double l = 2.0, double b = 2.0, double h = 2.0) {
9          cout << "Constructor called." << endl;
10         length = l;
11         breadth = b;
12         height = h;
13     }
14     double Volume() {
15         return length * breadth * height;
16     }
17     int compare(Box box) {
18         return this->Volume() > box.Volume();
19     }
20
21 private:
22     double length;    // Length of a box
23     double breadth;   // Breadth of a box
24     double height;    // Height of a box
25 };
26
27 int main(void) {
28     Box Box1(3.3, 1.2, 1.5);    // Declare box1
29     Box Box2(8.5, 6.0, 2.0);    // Declare box2
30
31     if(Box1.compare(Box2)) {
32         cout << "Box2 is smaller than Box1" << endl;
33     } else {
34         cout << "Box2 is equal to or larger than Box1" << endl;
35     }
36
37     return 0;
38 }
```