C++

Ders 1

C++ ve Nesne Yönelimli Programlamaya Giriş

Prof. Dr. M. Serdar ÇELEBİ, İTÜ

C++

İyi Bir Programın Özellikleri

- **Doğruluk (Correctness);** Verilen görevin tam olarak yerine getirilmesidir. Bir yazılımı yazmaya başlamadan önce yazılımdan beklenenlerin belirtilmesi gerekir. Yazılım ortaya çıktıktan sonra bu belirlenen özellikler tam olarak sağlanmalıdır.

- **Güvenilirlik (Robustness);** Beklenmedik nedenlerden dolayı programın çalışması kesilmemeli, yanlış işlemler yapmamalıdır. Program iyi yönde bile olsa kendinden beklenmeyen işlemler yapmamalıdır. Programcı hataları yüzünden kesintiye uğramaması için önlem almalıdır.

C++

İyi Bir Programın Özellikleri - II

- **Genişleyebilme (Extendibility);** İleri aşamalarda verilen görevlerin değiştirilmesi veya yenilerinin ilave edilmesi kolay olmalıdır. Bunun için basit tasarımlar yaparak karmaşık tasarımlardan uzak durmak gerekir.

- **Tekrar kullanım (Reusability)**; Yapılan tasarım şeklinin yazılan programın veya hiç olmassa modüllerin başka programlar içinde kullanılabilmesidir. Örneğin, bir konuda bir proje içinde kullanılan elamanların /veya modüllerin yeni bir projede tekrar kullanılabilmesi gerekir.

C++

İyi Bir Programın Özellikleri - III

- **Uygunluk (Compatibility);** farklı bilgisayar sistemlerinde aynı ortak özelliklere sahip olunması. Bunun için çeşitli standartların kabul edilip uygulanması gerekir: örnek olarak,
  - veri dosyası formatının uygunluğu
  - veri yapılarının uyumluluğu
  - menu, diyalog, resim, tuş gibi kullanıcı ara yüzünün (user interface) uyumluluğu

- **Kaynakların kullanımı (Efficiency);** bilgisayarın sahip olduğu ve ulaşabildiği tüm ekipmanları en iyi şekilde kullanmak. Eksik kullanmamak, kullanmadığı halde diğer programlara kullandırmamazlık etmemek.

C++

İyi Bir Programın Özellikleri - IV

- **Taşınabilirlik (Portability);** çoğu zaman bir yazılım ürünün geliştirildiği bilgisayar ortamından farklı bilgisayar ortamına taşınarak çalıştırılması gerekir. Bu iki şekilde olabilir:
  - **Kaynak uyumlu (Source Compatable);** yazılan programın yazıldığı bilgisayar sisteminden farklı bir sistemde tekrar derlenerek çalıştırılabilmesi.
  - **İkili kod uyumlu (Binary Compatable);** programın yazıldığı ortamda derlenip, icra edilebilir dosyası (executable file) elde edildikten sonra farklı bir ortama taşınarak çalıştırılması.

  Yazılım geliştirilmesi acısından asıl üzerinde durulması gereken kaynak kodun taşınabilirliğidir.

C++

İyi Bir Programın Özellikleri - V

- **Kontrol edilebilirlik (Verifiability);** bir yazılımın hatalı durumlar ile karşılaşması halinde programın devam etmesine engel olan ve hatta programı çalıştırmayan hatalar ortaya çıktığında kullanıcıya ve programcıya hatanın ne olduğu konusunda yeterli bilginin verilebilmesidir.

- **Bütünlük (Integrity);** istenmeyen erişimler ve değiştirmeler karşısında program, veri, dokümantasyon gibi yazılım birleşenlerinin korunması. Bir programcının veri veya index dosyalarının programın bilgisi dışında kaybolması programın çalıştığında bunu anlayıp rapor etmesi buna örnek olarak verilebilir.

C++

İyi Bir Programın Özellikleri - VI

- **Kolay kullananım (Easy use);** yazılım ürününü kullanan kişinin ürünü kolayca öğrenmesi, sorunsuz kullanabilmesi, sonuçlarını yorumlayabilmesi, hatalarını düzelterek işlerini fazla zahmete girmeden yapabilmesidir.

- **Beraber çalışma (Interoperability);** bir yazılım ürünün ihtiyaç duyduğu bir diğer yazılımı çağırabilmesi veya diğer bir yazılım tarafından çağrılabilmesi özelliğidir.

  Bu durumda iki program ard arda çalışma dışında bir birleri ile veri alışverişinde bulunabilmelidirler.

C++

Programlama Dilleri

- Makine Dili:

  Tüm komutlar sayılar şeklindedir.

```
+1300042774
+1400593419
+1200274027
```

- Assembler:

  Temel bilgisayar işlemleri kısaltmalarla ifade edilir.

```
LOAD BASEPAY
ADD OVERPAY
STORE GROSSPAY
```

- Yüksek Seviyeli Diller:

  Komutlar konuşma dili gibidir. Genel matematik notasyonu kullanılır.

```
grossPay = basePay + overTimePay
```

C++

C++'nin Tarihçesi

- C++, C'den geliştirilmiştir.
- ANSI C, C için dünyaca kabul edilen bir standarttır.
- C++, C'ye 'Nesneye Yönelik Programlama (OOP)' yetenekleri kazandırmıştır. Bu ise daha kolay anlaşılabilir ve düzeltilebilir bir program yazabilmeyi kolaylaştırır !

C++

C++'nin Tarihçesi

1979 (Mayıs)	1979 **C** with Classes ile ilgili çalışmalar başladı.
1979 (Ekim)	İlk "**C** with Classes" derleyicisi yazıldı.
1983 (Ağustos)	İlk **C++** derleyicisi
1983 (Aralık)	Dil **C++** ismini alıyor.
1989	ANSI X3J16 komitesi (daha sonra J16 ismini aldı) oluşturuldu.
1994	ANSI/ISO standart taslağı hazırlanıyor.
1996	2. Standart taslağı ANSI / ISO tarafından onaylandı.
1998	ISO **C++** dili standartları kabul ediliyor.

C++

C++'ın yapısı

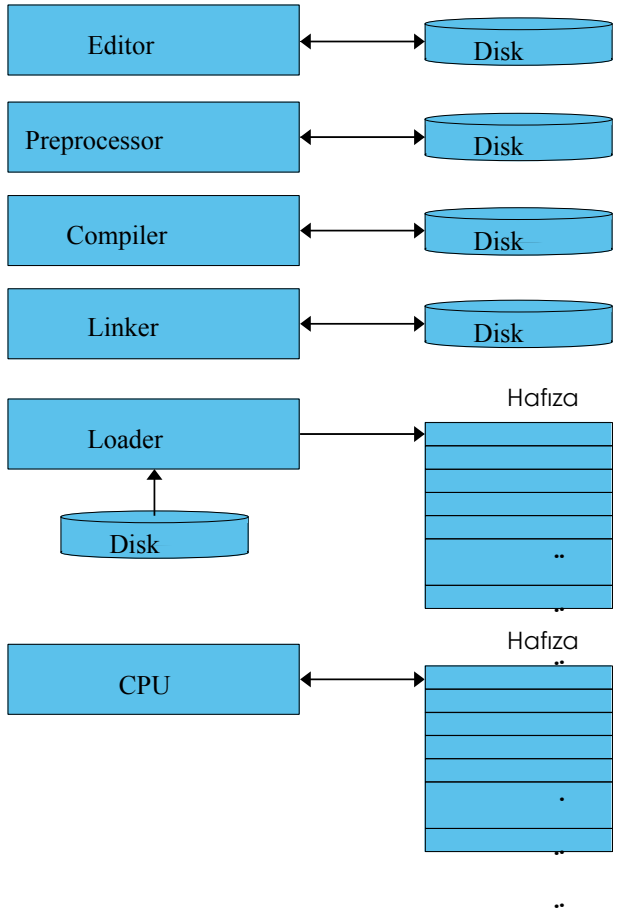
- C++ programları, **class** (sınıf) ve **function** (fonksiyon) denilen parçalardan oluşturulur.
- C++ Standart Kütüphanesi, tüm programcıların kullanabileceği zengin bir sınıf (class) ve fonksiyon (function) kolleksiyonu sunar.

C++

Bir Programın Aşamaları:

1. Yazım
2. Önişleme
3. Derleme
4. Bağlama
5. Yükleme
6. Çalıştırma

```
3      g++ -c test.cpp
4      g++ -o test test.o -lm
6      ./test
```



C++

C++ Programlamaya Giriş

- C ve C++ platformlar arasında taşınabilir kodlar üretir.
- C++, nesne yönelimli programlama yeteneklerinin yanısıra yapısal ve disiplinli programlamaya olanak veren özellikler sağlar.

C++

Bir C++ Programı

```
1  // Fig. 1.2: fig01_02.cpp
2  // A first program in C++
3  #include <iostream>
4
5  int main()
6  {
7     std::cout << "Welcome to C++!\n";
8
9     return 0;      // indicate that program ended successfully
10 }
```

Yorum Satırları

Önişlemci komutu

Fonksiyon deklarasyonu (özel main fonksiyonu)

Fonksiyon { ile başlar, } ile biter

Komutlar. Karekter dizisinin “ ” içine yazıldığına dikkat ediniz.

return ile fonksiyondan çıkılır.

C++

İlk Programımız: Bir yazı yazdırmak!

- std::cout

  -standart ‘output stream’ nesnesidir,

  -ekrana çıktı verir.

  -std:: ‘namespace’ in adını gösterir, ‘using’ kullanılırsa yazılması gerekmez.

- <<

  -stream ekleme operatörüdür.

  -sağındaki değeri ‘output stream’e gönderir.

  -std::cout << “Welcome to C++!\n”;

C++

İlk Programımız: Bir yazı yazdırmak!

- \ 'escape' karakteri özel karakterleri yazdırmak için kullanılır:

Escape İfadesi	Açıklama
\n	Yeni satır açar. İmleci yeni satırın başına taşır..
\t	Yatay tab. İmleçi bir sonraki tab noktasına taşır.
\r	Carriage return. İmleci satırın başlangıcına taşır. Ama yeni bir satır açmaz.
\a	'Beep' sesini çıkarır.
****	Backslash karakterini yazdırmak için kullanılır.
\"	Çift Tırnak yazdırmak için kullanılır.

C++

İlk Programımız: Bir yazı yazdırmak!

```
1  // Fig. 1.5: fig01_05.cpp
2  // Printing multiple lines with a single statement
3  #include <iostream>
4
5  int main()
6  {
7     std::cout << "Welcome\nto\n\nC++!\n";
8
9     return 0;   // indicate that program ended
10 }
```

```
Welcome
to

C++!
```

Bu ifade tek bir satır olmasına rağmen birden çok satır üretir

C++

Karşılaştırma Operatörleri

Standard Aritmetik **Eşitlik ve Karşılaştırma** operatörleri	C++ 'deki karşılığı	Örnek C++ ifadesi	İfadenin C++ 'deki anlamı
Karşılaştırma Operatörleri			
>	>	x > y	x, y 'den büyüktür
<	<	x < y	x, y 'den küçüktür
≥	>=	x >= y	x, y 'den büyük yada eşittir
≤	<=	x <= y	x, y 'den küçük yada eşittir
Eşitlik Operatörleri			
=	==	x == y	x, y 'e eşittir.
≠	!=	x != y	x, y 'e eşit değildir.

C++

Nesne Yönelimli Programlama (OOP)

- Dünyayı düşündüğümüz doğallıkla programlama yapabilmemizi sağlar.
- Nesne yönelimli programlama tekniği kullanarak geliştirilmiş programlar daha güvenilir, dayanıklı ve kolay geliştirilebilirler.
- Nesne'yi, belirlenmiş bir işlevi yerine getirebilecek altyapıyı hazırlayan, bu amaçla çeşitli fonksiyonlar içeren bir yapı olarak tanımlayabiliriz.

C++

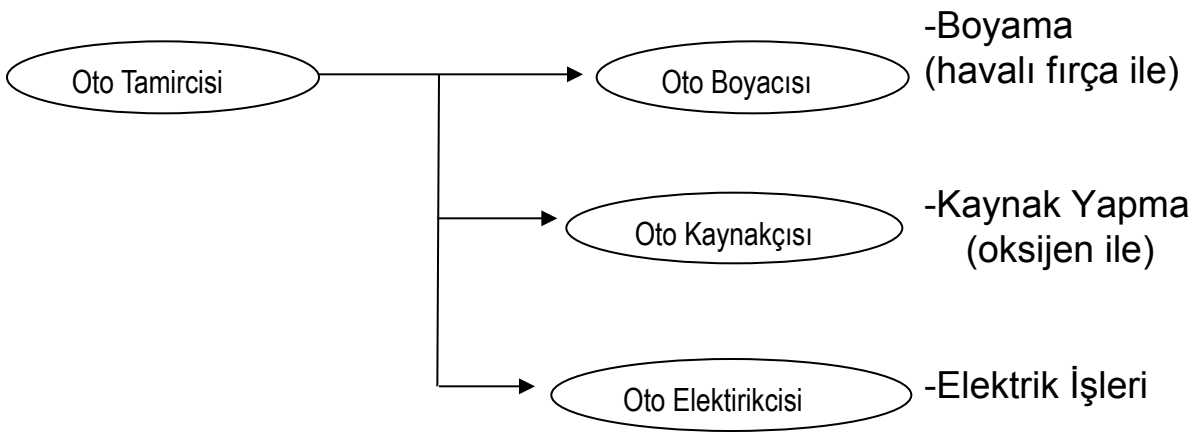
Nesne Nedir?

- Gerçek dünyadaki nesnelere benzeyen tekrar kullanılabilir yazılım parçalarıdır.
- Tek başlarına bir anlamları vardır: *zaman, ses, video nesnesi, boyacı, sekreter* gibi.
- Daha anlaşılır, değiştirilmesi daha kolay bir yapı sunarlar.
- Belirlenmiş işlevleri yerine getiren, bunun içinde çeşitli fonksiyonlar içeren bu yapı değişkenleri de bünyesinde bulundurabilir. Fakat asıl, işlevini belli edecek fonksiyonları bünyesinde bulundurmasıdır. Bu özelliğe Paketleme (Encapsulation) adı verilir.
- Paketlenecek fonksiyonların nasıl bir işlev göstereceği belirtilmeksizin, sadece nasıl kullanılacağının belirtilmesine Soyutlama (Abstraction) adı verilir.
- Paketleme ve Soyutlama bir nesneyi belli etmek için yeterli iki özelliktir.

C++

Nesne Nedir? (Türeme & Kalıtım)

Bir oto tamircisi nesnesi düşünelim. Bu oto tamircisinin üç oğlu olduğunu ve bunların her birini yetiştirdikten sonra , birer konuda uzmanlaşmalarını sağladığını düşünelim.



Burada Oto Boyacısı, Oto Kaynakçısı ve Oto Elektrikcisi türeyen nesnelerdir. Oto tamircisi ise taban nesnedir. Türeyen nesneler taban nesnenin özelliklerini göstereceklerdir. Oto boyayacaklar, kaynak yapacaklar, karbüratör ayarlayacaklar. Yeni nesne kendisine yeni özellikler katabileceği gibi devraldığı özellikleri de geliştirebilir. Bu özelliğe Türeme (Derivation), özellikleri devralmaya da Miras alma veya Kalıtım (Inheritance) adı verilir.

C++

Nesne Nedir? (Benzerlik)

Oto tamircisi örneğine devam edersek, Burada üç önemli özellik göze çarpar:

1) Türeyen nesneler taban nesnenin özelliklerini koruyarak devam ettirip kullanabilirler
2) Türeyen nesneler türedikleri nesnelerin (taban nesnelerin) özelliklerini değiştirebilirler
3) Türeyen nesneler yeni özellikler kazanabilir veya mevcut özellikleri kaybedebilirler

O zaman bir sistem içerisinde ister bir nesneden türemiş nesneler bulunsun, isterse bir birinden farklı nesneler bulunsun, bu nesnelerin benzer özellikleri olabilir ve bu özellikler aynı isim ile anılır. Buda benzer isimde fakat farklı nesnelerin üyesi olan üyelerin ortaya çıkmasına neden olur. Bu durum nesneler arasındaki benzerlikleri yansıtır. Bu duruma Çoklubenzeşim (Polymorphism) adı verilir.

C++

Nesne

- Özellikleri (Attributes, Properties) vardır: renk, şekil, boyut, genişlik, yükseklik vb.
- Davranışları (Behaviors, Actions) vardır: her nesnenin kendine özgü davranışları vardır. Top yuvarlanır, ağaç yanar.
- Yeni nesneler Mirasla (Inheritance) eski (parent) nesnelerin bazı özellik ve yöntemlerine sahip olurlar.

C++

Nesne II

- Bilgiler Gizlenebilir: Nesnelerin iş görmesi için diğer nesneler hakkında bilgi sahibi olması şart değildir.
- Soyutlama (Abstraction) sayesinde bir nesne, spesifik tanımlamalar ve sadece o nesneye ait olabilecek detaylardan kaçınarak, ayrıntılarda boğulmadan kullanılabilir.
- Sınıflar (Class) nesnelerin temelini oluşturulurlar.

C++

Yapısal Programlama (Structured-Programming)

- Anlaması, test etmesi, değiştirmesi kolay programlar yazmayı teşvik eder ve kolaylaştırır.
- Her zaman sadece bir girişi, bir çıkışı olan program yapıları kullanır.
- **Kurallar:**

  1) En basit şema (flowchart) ile başla,

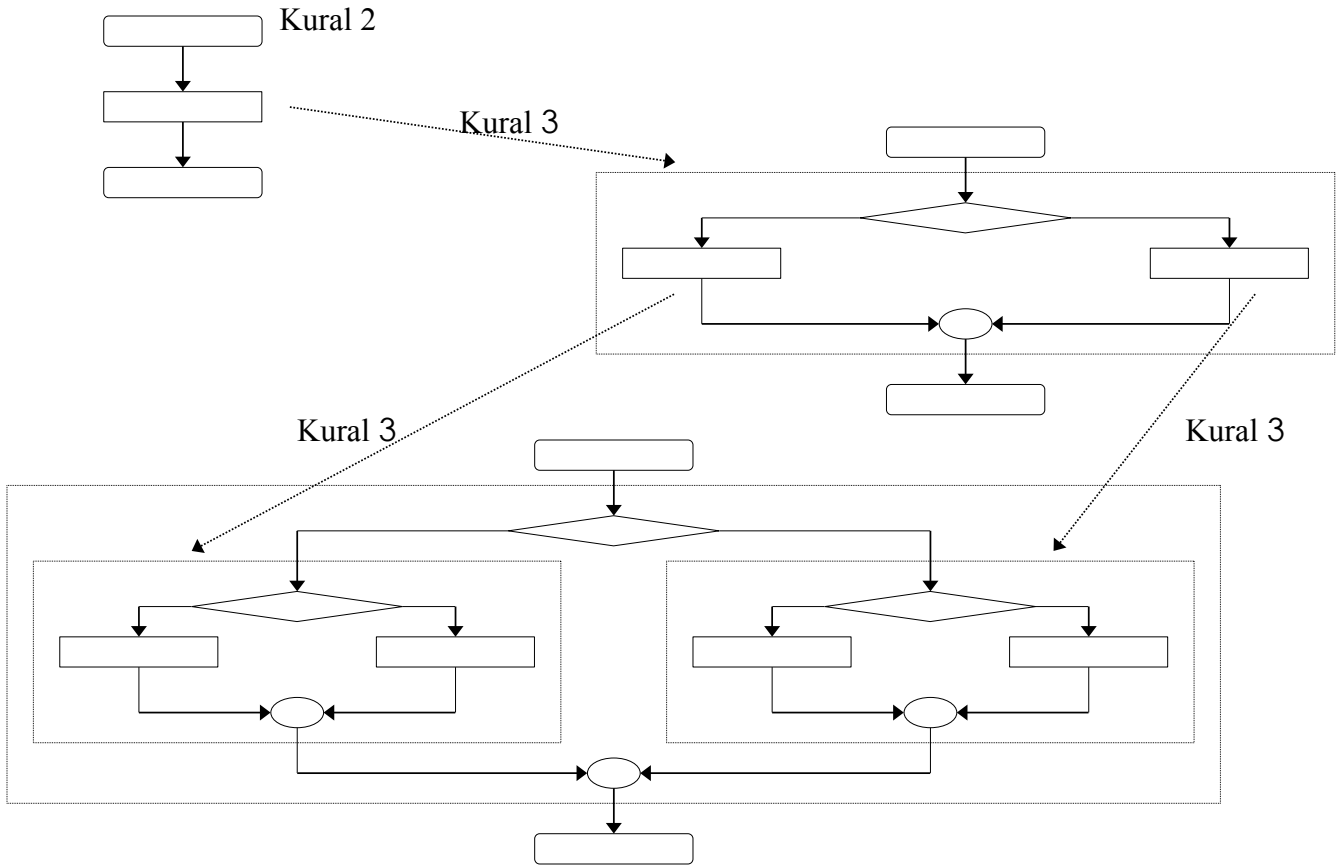
  2) Her hangi bir işlem birbirini takip eden iki işlem şeklinde yazılabilir.

  3) Her hangi bir işlem, bir kontrol yapısı (döngü, koşul, komut bloğu vb.) ile değiştirilebilir.

  4) Kural 2 ve 3, birçok kere tekrarlanabilir.

C++

Yapısal Programlama Kurallarının uygulanışı



C++

Yapısal Programlama

Tüm programlar:

- Komut Bloğu,
- Koşul / Seçim (if, if/else, switch),
- Döngü (for, while, do/while)

Parçalarına bölünebilir, yada diğer bir değişle bu parçalardan oluşturulabilir.

C++

Nesne Yönelimli Programlama (OOP)

- Veri ve Fonksiyonlar Sınıf (Class) paketlerinin içine saklanır.
- Class'lar mimari planlar gibidir.
  - Tekrar tekrar kullanılabilirler.
  - Class (plan) kullanılarak yeni nesneler (evler) oluşturulur.
  - Planlarda değil, evlerde otururuz.

C++

Nesne Yönelimli Programlama (OOP)

- Nesne Yönelimli Programlamada
  - Veri (attributes) ve
  - Fonksiyonlar (behavior)

  Sınıf olarak adlandırılan paketler içinde korunurlar, ve birbirleri ile son derece sıkı ilişki içindedirler.

C++

Struct Yapısı

- Diğer data tiplerini birleştirerek, yeni bir data tipi oluşturur.

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

Struct adı

Struct üyeleri

C++

Struct Yapısı

- Bir struct yapısındaki her üyenin adı farklı olmalıdır.
- İki farklı struct yapısı aynı üyelere sahip olabilir.
- Struct tanımı mutlaka ‘;’ ile bitirilir.
- Kendisini üye olarak alan bir struct yazılabilir ve genelde liste, stack vb. yapılar programlamada kullanılır.

C++

Struct Yapısı

- Struct yeni bir değişken tipi oluşturur, yani bu ‘yeni’ tipte değişkenler kullanılarak başka değişkenler tanımlayabilirsiniz:

```
Time timeObject, timeArray[ 10 ],
   *timePtr, &timeRef =timeObject;
```

- Normal üyelere erişirken ‘.’, pointer üyelere erişirken ‘->’ operatörleri kullanılır.

C++

Struct Yapısı

- **`timeObject`** struct yapısının **`hour`** üyesine erişirken:

```
cout << timeObject.hour;
timePtr = &timeObject;
cout << timePtr->hour;
```

- Burada '**`timePtr->hour`**' ile '**`(*timePtr ).hour`**' tamamen aynıdır.

C++

Örnek (struct)

```
1  // Fig. 6.1: fig06_01.cpp
2  // Create a structure, set its members, and print it.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  struct Time {     // structure definition
9     int hour;      // 0-23
10    int minute;    // 0-59
11    int second;    // 0-59
12 };
13
14 void printMilitary( const Time & );  // prototype
15 void printStandard( const Time & );  // prototype
16
```

C++

Örnek (devam)

```
17 int main()
18 {
19    Time dinnerTime;    // variable of new type Time
20
21    // set members to valid values
22    dinnerTime.hour = 18;
23    dinnerTime.minute = 30;
24    dinnerTime.second = 0;
25
26    cout << "Dinner will be held at ";
27    printMilitary( dinnerTime );
28    cout << " military time,\nwhich is ";
29    printStandard( dinnerTime );
30    cout << " standard time.\n";
31
```

C++

Örnek (devam)

```
32    // set members to invalid values
33    dinnerTime.hour = 29;
34    dinnerTime.minute = 73;
35
36    cout << "\nTime with invalid values: ";
37    printMilitary( dinnerTime );
38    cout << endl;
39    return 0;
40 }
41
42 // Print the time in military format
43 void printMilitary( const Time &t )
44 {
45    cout << ( t.hour < 10 ? "0" : "" ) << t.hour << ":"
46         << ( t.minute < 10 ? "0" : "" ) << t.minute;
47 }
```

C++

Örnek (devam)

```
49 // Print the time in standard format
50 void printStandard( const Time &t )
51 {
52    cout << ( ( t.hour == 0 || t.hour == 12 ) ?
53                12 : t.hour % 12 )
54         << ":" << ( t.minute < 10 ? "0" : "" ) << t.minute
55         << ":" << ( t.second < 10 ? "0" : "" ) << t.second
56         << ( t.hour < 12 ? " AM" : " PM" );
57 }
```

```
Dinner will be held at 18:30 military time,
which is 6:30:00 PM standard time.

Time with invalid values: 29:73
```

C++

Time Class (Sınıf) Deklarasyonu

- Sınıf '{ }' işaretleri arasında tanımlanır. Tanım ';' işareti ile sonlandırılmalıdır.
- Örnek:

```
1  class Time {
2  public:
3     Time();
4     void setTime( int, int, int );
5     void printMilitary();
6     void printStandard();
7  private:
8     int hour;      // 0 - 23
9     int minute;    // 0 - 59
10    int second;    // 0 - 59
11 };
```

Public: yada **Private:** erişim kontrolü içindir.

setTime, **printMilitary**, ve **printStandard** üye fonksiyonlardır.

Time ise **constructor**'dür.

C++

Üye (Member) Erişim Kısıtlamaları

- Sınıf' lar kendi üyelerine (veri ve fonksiyonlarına) dışarıdan erişimi sınırlayabilirler.
- **Public**: sınıf' a erişilen her yerde bu üyelere de erişilebilir.
- **Private**: sadece sınıf' ın kendi üye fonksiyonları bu üyeye erişebilir.
- **Protected**: private' a benzer. Daha sonra ayrıntılı olarak anlatılacak.

C++

Constructor

- Sınıf yapısının özel bir fonksiyonudur. Sınıf' la aynı isimdedir.
- Sınıf' tan bir nesne oluşturulurken çalıştırılır. Sınıf üyelerini hazırlar.
- Geri dönüş değeri olmaz ama parametre alabilir.
- Bir kez sınıf tanımlanınca, bir değişken tipi gibi kullanılır. Dizi, pointer yada normal bir değişken tanımlanabilir.

C++

Örnek (Sınıf)

```
1  // Fig. 6.3: fig06 03.cpp
2  // Time class.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  // Time abstract data type (ADT) definition
9  class Time {
10 public:
11    Time();                          // constructor
12    void setTime( int, int, int ); // set hour, minute, second
13    void printMilitary();            // print military time format
14    void printStandard();            // print standard time format
15 private:
16    int hour;     // 0 - 23
17    int minute;   // 0 - 59
18    int second;   // 0 - 59
19 };
20
```

C++

Örnek (devam)

```
21 // Time constructor initializes each data member to zero.
22 // Ensures all Time objects start in a consistent state.
23 Time::Time() { hour = minute = second = 0; }
24
25 // Set a new Time value using military time. Perform validity
26 // checks on the data values. Set invalid values to zero.
27 void Time::setTime( int h, int m, int s )
28 {
29    hour = ( h >= 0 && h < 24 ) ? h : 0;
30    minute = ( m >= 0 && m < 60 ) ? m : 0;
31    second = ( s >= 0 && s < 60 ) ? s : 0;
32 }
33
34 // Print Time in military format
35 void Time::printMilitary()
36 {
37    cout << ( hour < 10 ? "0" : "" ) << hour << ":"
38         << ( minute < 10 ? "0" : "" ) << minute;
39 }
```

: : operatörü class'ların fonksiyonlarına erişmek için kullanılıyor.

C++

Örnek (devam)

```
41 // Print Time in standard format
42 void Time::printStandard()
43 {
44    cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 )
45         << ":" << ( minute < 10 ? "0" : "" ) << minute
46         << ":" << ( second < 10 ? "0" : "" ) << second
47         << ( hour < 12 ? " AM" : " PM" );
48 }
49
50 // Driver to test simple class Time
51 int main()
52 {
53    Time t;  // instantiate object t of class Time
54
55    cout << "The initial military time is ";
56    t.printMilitary();
57    cout << "\nThe initial standard time is ";
58    t.printStandard();
59
```

C++

Örnek (devam)

```
60    t.setTime( 13, 27, 6 );
61    cout << "\n\nMilitary time after setTime is ";
62    t.printMilitary();
63    cout << "\nStandard time after setTime is ";
64    t.printStandard();
65
66    t.setTime( 99, 99, 99 );  // attempt invalid settings
67    cout << "\n\nAfter attempting invalid settings:"
68         << "\nMilitary time: ";
69    t.printMilitary();
70    cout << "\nStandard time: ";
71    t.printStandard();
72    cout << endl;
73    return 0;
74 }
```

C++

Örnek (Ekran Çıktısı)

```
The initial military time is 00:00
The initial standard time is 12:00:00 AM

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

After attempting invalid settings:
Military time: 00:00
Standard time: 12:00:00 AM
```

C++

Destructor Fonksiyonlar

- 'Destructor' fonksiyon sınıf isminin önüne '~' işareti konularak isimlendirilir.
- Nesne yok edilmeden önce yapılması gerekenleri yapar.
- Fonsiyon aşırı-yüklemesi (overloading) yapılamaz. Parametresi olamaz.

```
class time {
        public:
        time();         // constructor
        ~time();        // destructor
        };
```

C++

'Binary scope resolution' Operatorü (::)

- Sınıf' ın üyelerine erişirken sınıf isminin de kullanılabilmesini sağlar.
- Başka sınıf' ların da aynı isimli üyeleri olabileceği için gereklidir.

```
dönüştipi ClassAdı::ÜyeFonksiyonAdı( )
        {
        ...
        }
```

C++

Erişim Alanı

- **'Class scope'** : Bir sınıfın veri ve fonksiyon üyeleri bu sınıf alanına aittir.
- **'File scope'** : Üye olmayan fonksiyonlar bu alanda tanımlanır.
- Erişim alanı (scope) içinde iken üye fonksiyonlara adları ile erişilebilir.
- Erişim alanı dışında ise üye fonksiyonlara başka bir nesne yada pointer aracılığı ile ulaşılır.

C++

Erişim Alanı - II

- **'Function Scope'** : Bir üye fonksiyon içinde tanımlanan değişkenler sadece tanımlandıkları fonksiyon tarafından bilinirler. Fonksiyon çağırıldığında oluşturulup, fonksiyondan çıkışta yok edilirler.
- Üyelere erişim aynı struct yapısındaki gibidir. Normalde '**.**', Pointer'larla ise '**->**' operatörleri kullanılır:

  t.hour yada ***timePtr->hour*** gibi.

C++

Örnek ('.' ve '->')

```
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 // Simple class Count
11 class Count {
12 public:
13    int x;
14    void print() { cout << x << endl; }
15 };
16
17 int main()
18 {
```

public üye değişkenler çok nadiren kullanılır. Genelde gizlenen değişkenlere public fonksiyonlar ile erişim tercih edilerek bilgi gizlenir. Bu sayede nesne soyutlanır.

C++

Örnek (devam)

```
19    Count counter,                // create counter object
20          *counterPtr = &counter, // pointer to counter
21          &counterRef = counter;  // reference to counter
22
23    cout << "Assign 7 to x and print using the object's name: ";
24    counter.x = 7;          // assign 7 to data member x
25    counter.print();        // call member function print
26
27    cout << "Assign 8 to x and print using a reference: ";
28    counterRef.x = 8;       // assign 8 to data member x
29    counterRef.print();     // call member function print
30
31    cout << "Assign 10 to x and print using a pointer: ";
32    counterPtr->x = 10;     // assign 10 to data member x
33    counterPtr->print(); // call member function print
34    return 0;
35 }
```

```
Assign 7 to x and print using the object's name: 7
Assign 8 to x and print using a reference: 8
Assign 10 to x and print using a pointer: 10
```

C++

‘Interface’ - ‘Implementation’ Bölümlerinin Ayrılması

- Sınıf yapısının tanımlanması bir başlık dosyasında (header file) yapılır.
- Üye fonksiyonlar, program dosyasında tanımlanır.
- Bu programların değiştirilmesini kolaylaştırır.

C++

Örnek (Başlık Dosyası)

time1.h →

```
5  // prevent multiple inclusions of header file
6  #ifndef TIME1_H
7  #define TIME1_H
8
9  // Time abstract data type
10 class Time {
11 public:
12    Time();                          // constructor
13    void setTime( int, int, int ); // set hour, minute, second
14    void printMilitary();            // print military time format
15    void printStandard();            // print standard time format
16 private:
17    int hour;      // 0 - 23
18    int minute;    // 0 - 59
19    int second;    // 0 - 59
20 };
21
22 #endif
```

eğer **time1.h** (**TIME1_H**) tanımlı değilse (**#ifndef**) onu tanımla (**#define TIME1_H**). Eğer **TIME1_H** zaten tanımlı ise **#endif**'ten öncesi çalıştırılmaz.

Bu aynı header file'ın tekrar yüklenmesini önler.

C++

Örnek (Program Dosyası)

```
23 // Fig. 6.5: time1.cpp
24 // Member function definitions for Time class.
25 #include <iostream>
26
27 using std::cout;
28
29 #include "time1.h"
30
31 // Time constructor initializes each data member to zero.
32 // Ensures all Time objects start in a consistent state.
33 Time::Time() { hour = minute = second = 0; }
34
35 // Set a new Time value using military time. Perform validity
36 // checks on the data values. Set invalid values to zero.
37 void Time::setTime( int h, int m, int s )
38 {
39    hour   = ( h >= 0 && h < 24 ) ? h : 0;
40    minute = ( m >= 0 && m < 60 ) ? m : 0;
41    second = ( s >= 0 && s < 60 ) ? s : 0;
42 }
```

#include ile sınıf ın tanımlandığı dosyayı (time1.h) yüklüyoruz!

C++

Örnek (devam)

```
43
44 // Print Time in military format
45 void Time::printMilitary()
46 {
47    cout << ( hour < 10 ? "0" : "" ) << hour << ":"
48         << ( minute < 10 ? "0" : "" ) << minute;
49 }
50
51 // Print time in standard format
52 void Time::printStandard()
53 {
54    cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 )
55         << ":" << ( minute < 10 ? "0" : "" ) << minute
56         << ":" << ( second < 10 ? "0" : "" ) << second
57         << ( hour < 12 ? " AM" : " PM" );
58 }
```

Fonksiyonların içeriği program dosyasında yazılı!

C++

Üyelere Erişim

- **Public**: bu bölümdeki veri ve fonksiyonlara erişilebilir.
- **Private**: bu bölümdekilere sadece üye yada **friend** fonksiyonlar üzerinden erişilebilir.
- Hiçbiri yazılmazsa her şey **private** kabul edilir.

C++

Örnek (scope)

```
1  // Fig. 6.6: fig06_06.cpp
2  // Demonstrate errors resulting from attempts
3  // to access private class members.
4  #include <iostream>
5
6  using std::cout;
7
8  #include "time1.h"
9
10 int main()
11 {
12    Time t;
13
14    // Error: 'Time::hour' is not accessible
15    t.hour = 7;
16
17    // Error: 'Time::minute' is not accessible
18    cout << "minute = " << t.minute;
19
20    return 0;
21 }
```

C++

Örnek (Ekran Çıktısı)

- Private olarak tanımlanan hour değişkenine erişim ve değiştirme çabası derleme aşamasında hataya sebep olur:

```
Compiling...
Fig06_06.cpp
D:\Fig06_06.cpp(15) : error C2248: 'hour' : cannot access private
member declared in class 'Time'
D:\Fig6_06\time1.h(18) : see declaration of 'hour'
D:\Fig06_06.cpp(18) : error C2248: 'minute' : cannot access private
member declared in class 'Time'
D:\time1.h(19) : see declaration of 'minute'
Error executing cl.exe.

test.exe - 2 error(s), 0 warning(s)
```

C++

Hizmet-Erişim Fonksiyonları

- **Hizmet fonksiyonları:** Nesnenin yapması gereken işleri bu özel (private) fonksiyonlar yaparlar ve erişim fonksiyonlarına destek verirler.
- **Erişim fonksiyonları:** Nesnenin özel (private) fonksiyon ve verilerine bir ara yüz oluşturan ve bu erişimin güvenli ve kolay olmasını sağlayan fonksiyonlardır.

C++

Örnek (Erişim Fonk.)

```
// Fig. 6.7: salesp.h
// SalesPerson class definition
// Member functions defined in salesp.cpp
#ifndef SALESP_H
#define SALESP_H
class SalesPerson {
public:
  SalesPerson();                    // constructor
  void getSalesFromUser(); // get sales figures from keyboard
  void setSales( int, double ); // User supplies one month's
                                    // sales figures.
  void printAnnualSales();

private:
  double totalAnnualSales();   // utility function
  double sales[ 12 ];           // 12 monthly sales figures
};
#endif
```

C++

Örnek (Erişim Fonk.)

```
87 // Fig. 6.7: fig06_07.cpp
88 // Demonstrating a utility function
89 // Compile with salesp.cpp
90 #include "salesp.h"
91
92 int main()
93 {
94    SalesPerson s;          // create S
95
96    s.getSalesFromUser();  // note simple sequential code
97    s.printAnnualSales();  // no control structures in main
98    return 0;
99 }
```

Bu erişim fonksiyonları kullanıcıdan data alır ve yazıcıda basar (**`getSalesFromUser`** and **`printAnnualSales`**). Toplam satış hesabını aslında hizmet programları yapar ama kullanıcı açısından bu önemsizdir.

Program bu haliyle oldukça sadedir.

C++

Constructor' lar

- Sınıf üyelerini hazırlarlar. Ancak zorunlu değildir.
- Sınıf' la aynı adlıdırlar ve geri dönüş değerleri yoktur.
- Sınıf tanımlanırken default parametreler tanımlanabilir.
- Yada nesne tanımlanırken parametre geçilebilir:

  sınıf-adı nesne-adı(parametreler,…);

C++

Örnek (Constructor'lar)

```
5  // preprocessor directives that
6  // prevent multiple inclusions of header file
7  #ifndef TIME2_H
8  #define TIME2_H
9
10 // Time abstract data type definition
11 class Time {
12 public:
13    Time( int = 0, int = 0, int = 0 );  // default constructor
14    void setTime( int, int, int ); // set hour, minute, second
15    void printMilitary();          // print m
16    void printStandard();          // print s
17 private:
18    int hour;     // 0 - 23
19    int minute;   // 0 - 59
20    int second;   // 0 - 59
21 };
22
23 #endif
```

constructor prototipinde üç private değişkene default değer atanırken değişken ismi belirtilmediğine dikkat edin!

Yazılmazsa tanımlandıkları sırada atanacaklarından isim şart değildir.

C++

Örnek (Constructor'lar)

```
64 #include <iostream>
65
66 using std::cout;
67 using std::endl;
68
69 #include "time2.h"
70
71 int main()
72 {
73    Time t1,           // all arguments defaulted
74         t2(2),        // minute and second defaulted
75         t3(21, 34),   // second defaulted
76         t4(12, 25, 42), // all values specified
77         t5(27, 74, 99); // all bad values specified
78
79    cout << "Constructed with:\n"
80         << "all arguments defaulted:\n   ";
81    t1.printMilitary();
82    cout << "\n   ";
83    t1.printStandard();
```

Nesnelerin nasıl başlatıldığına dikkat edin:
Constructor ObjectName **(***value1,value2…***) ;**
Eğer yeterince değer girilmiyorsa eksik değerlerin sağ taraftakiler olduğu kabulü ile default değerlerle eksik tamamlanır.

C++

Örnek (Constructor'lar)

```
85     cout << "\nhour specified; minute and second defaulted:"
86          << "\n   ";
87     t2.printMilitary();
88     cout << "\n   ";
89     t2.printStandard();
90
91     cout << "\nhour and minute specified; second defaulted:"
92          << "\n   ";
93     t3.printMilitary();
94     cout << "\n   ";
95     t3.printStandard();
96
97     cout << "\nhour, minute, and second specified:"
98          << "\n   ";
99     t4.printMilitary();
100    cout << "\n   ";
101    t4.printStandard();
102
103    cout << "\nall invalid values specified:"
104         << "\n   ";
```

C++

Örnek (Constructor'lar)

```
105    t5.printMilitary();
106    cout << "\n    ";
107    t5.printStandard();
108    cout << endl;
109
110    return 0;
111 }
```

```
 OUTPUT
Constructed with:
all arguments defaulted:
   00:00
   12:00:00 AM
hour specified; minute and second defaulted:
   02:00
   2:00:00 AM
hour and minute specified; second defaulted:
   21:34
   9:34:00 PM
hour, minute, and second specified:
   12:25
   12:25:42 PM
all invalid values specified:
   00:00
   12:00:00 AM
```

Sadece hour girilince minute ve second'a default değeri olan 0 atanır.