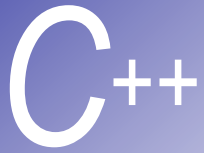


C++

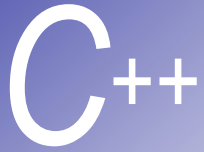
Ders 7

Hata Denetimi ve Dosya İşlemleri



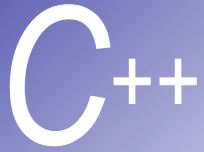
Stream'ler

- Stream
 - Byte' ların dizisi şeklinde bilginin transferidir.
- I/O Operasyonları:
 - Giriş: Bir giriş cihazından (klavye, disk drive, disk sürücü, network bağlantısı) ana belleğe akan stream.
 - Çıkış: Ana bellekten bir çıkış cihazına (ekran, yazıcı, disk sürücü , network bağlantısı) akan bilgi transferidir.



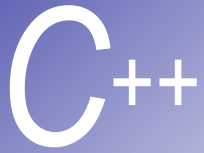
Giriş I

- Hata kontrolü yaptırılırsa hatayı görmek kolaydır:
 - Uygulamayı okuyup nasıl çalıştığını anlamak zor
- İstisna yakalama (Exception Handling)
 - programı anlaşılır, sağlıklı, hata toleranslı yapar
 - C++ error handling kodunu “main line” dan çıkarır
- Genel hatalar
 - **New**’ in memory tahsisi yapmaması
 - Diziyi taşıracak şekilde atama yapma
 - Geçersiz fonksiyon parametresi kullanma



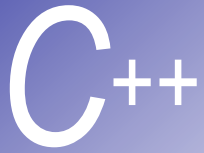
Giriş II

- İstisna yakalama – hata oluşmadan hatayı tespit eder
 - Tanımsızlıktan dolayı oluşan hatayı önler
(Sıfıra bölüm gibi..)
 - Asynchronous hatasını önlemez
 - Sistem hatayı düzeltebildiği sürece kullanılır
 - Hatanın nereden olduğundan çok değişik yerdeki hatalar için kullanılır
 - Program hatalardan temizlenemediğinde, onu durdurmak için kullanılır
- İstisna yakalama program kontrolü için kullanılmamalıdır
 - Optimal değildir, programın performansını düşürür



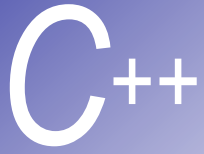
Giriş III

- İstisna yakalama hata toleransını artırır
 - error-processing kod yazması daha kolay
 - Ne tür hataların yakalanacağı belirlenmeli
- Birçok program yalnız single thread' leri destekler
 - Bu teknik multi threaded işletim sistemlerine de uygulandığı olmuştur (Windows NT, Unix)
 - Ne tür hataların yakalanacağı belirlenmeli
- İstisna Yakalama fonksiyon ve blokların hata kontrolü için bir yoldur



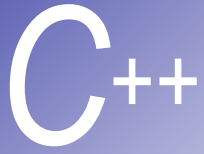
İstisna Yakalamamanın Kullanılması Geren Durumlar

- Hata yakalama şu durumlarda kullanılmalıdır:
 - Beklenmeyen süreç hatalarında
 - Direk kullanılmayan bileşenlerde hata oluşumunda
 - Yaygın kullanılan bileşenler (libraries, classes, functions) için kendi süreçleri çalıştırılmalıdır
 - Büyük projeler eş zamanlı hata işleme gerektirir.



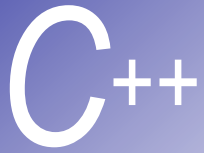
Diğer Hata yakalama Teknikleri (I)

- **assert** kullanımı
 - **assert** kullanımında hata olursa program sonlandırılır
- İstisna'ları yok saymak
 - Bu teknik ticari programlarda kullanılmamalıdır
 - cout, hangi türden bir verinin çıktı olacağını biliyor
 - Programdan abort ile çıkmak
 - Ölümcül olmayan hatalar görünebilir hale getirilebilir
 - Hatalar görüntülenirken gizli kaynaklar görüntülenmemelidir
 - Programa hata belirteçleri koymak
 - Hata belirteçleri istenildiği gibi hataları göstermeyebilir



Diğer Hata yakalama Teknikleri (II)

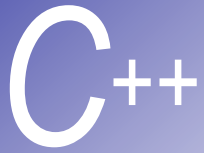
- Hata oluşabilecek durumlar için test yapmak
 - Hata oluştuğunda `exit` ile program sonlandırılmalı
 - Program sonlanırken hatayı belirtmeli
- `setjump` ve `longjump` kullanılabilir
 - `<csetjmp>` başlık dosyasında
 - Destructor kullanmadan tahsisat serbest bırakılmamalı
- Özel hatalar
 - `new` tahsisatta başarısız olursa `new_handler` ile sorun çözülmeli



C++' in Hata Yakalama

Temeli: `try`, `throw`, `catch`

- İstisna nesnenin bulunduğu hatayı **`throw`** yapabilir
 - Nesne genellikle karakter string' i dir
 - İstisna yakalayıcı varsa hatayı yakalar
 - Diğer durumlarda program sonlanır
 - newline character basar (imleci alt satıra götürür)
- Format
 - **`try`** blokları ile hata bulunduran kodlar ayrılabilir
 - **`catch`** blokları ile bölünüp her bloğun bir hata yakalayıcı bulundurulabilir
 - Hata durumu olursa ve parametre **`catch`** bloğuna uyarsa **`catch`** kodu çalıştırılabilir
 - Hata durumu yoksa hata yakalayıcı öncesini tarayıp kontrol edebilir
 - Hata olduğu yerde **`throw`** point bulundurulabilir

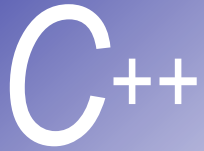


Bir Exception' i throw Yapma

- **throw** – bir hatanın olduğunu gösterir
 - Genellikle bir operand alır (bazen sıfır)
 - Eğer operand nesne ise, bir istisna nesnesi çağırılır
 - Şartlı ifade `throw` yapılabilir
 - **try** bloğu bir hata çıkarabilir
 - Ayrılmış istisna yakalayıcılar tarafından hata yakalanabilir
 - Mevcut olan kontroller blokları deneyip onlara **catch** handler gönderebilir
 - Örnek:

```
if ( denominator == 0 )  
    throw DivideByZeroException();
```

 - throws a `DivideByZeroException` object
- Hatanın programı sonlandırması gerekmez
 - Bununla birlikte hatanın bulunduğu bloklar sonlandırılabilir



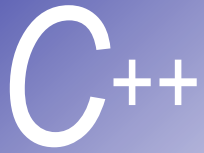
İstisna Yakalama (I)

- **catch** blokları içinde hata yakalayıcılar
 - Format: `catch(exceptionType parameterName) {
 exception handling code
}`
 - Argument tipi **throw** tipine uyarsa catching işlemi olur
 - Catching işlemi olmazsa **abort**' u içeren **terminate** çağırılır

Örnek:

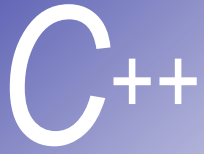
```
catch ( DivideByZeroException ex) {  
    cout << "Exception occurred: " << ex.what() << '\n'  
}
```

- `DivideByZeroException` tipi hataları yakalar
- Bütün hatalar yakalanabilir
 `catch(. . .)` –Türü ve parametre bilinmesede bütün exceptionları yakalayabilir



İstisna Yakalama (II)

- Thrown nesneye uygun handler yoksa:
 - Bir sonraki `try` blokta aranır
 - Eğer hala bulunamamışsa `terminate` çağırılır
 - Bulunmuşsa, son `catch` bloktan sonrasına bakılır
 - Eğer uygun birkaç tane handler varsa ilk bulunan çalıştırılır
- `catch` parametresi thrown nesneye şu durumlarda uyar:
 - `Catch parametresi` thrown nesnesi public kök-sınıfı olduğu zaman
 - `catch` handler `catch (...)` olduğunda
 - thrown `const` nesneler `const` türden bir parametreye sahip olduğunda
 - `catch` parametresi kök-sınıf pointer/ referans türünde olduğunda ve thrown nesne türetilmiş-sınıf işaretçisi/ referans türden türetildiğinde
- döngüleri yapmak için popüler bir yoldur
`while (cin >> grade)`
 - eğer EOF ile karşılaşırsa çıkarma (extraction) 0 (yanlış) döndürür ve döngü biter



İstisna Yakalama (III)

- Serbest bırakılmamış kaynaklar
 - hata temizlenirken kaynak tahsisi yapılır
 - **catch** handler new tarafından yapılan tahsisatı silmeli ve açık dosyaları kapamalı
- **catch** handler hataları temizleyebilir
 - hatalara sadece **try** blok'ları dışında işlem yapılabilir

C++

Basit bir İstisna Yakalama Örneği

```

1 // Fig. 13.1: fig13_01.cpp
2 // A simple exception handling example
3 // Checking for a divide-by-zero exception.
4 #include <iostream>
5
6 using std::cout;
7 using std::cin;
8 using std::endl;
9
10 // Class DivideByZeroException to be used in exception
11 // handling for throwing an exception on a division by zero.
12 class DivideByZeroException {
13 public:
14     DivideByZeroException()
15         : message( "attempted to divide by zero" ) { }
16     const char *what() const { return message; }
17 private:
18     const char *message;
19 };
20
21 // Definition of function quotient. Demonstrates
22 // an exception when a divide-by-zero exception is encountered
23 double quotient( int numerator, int denominator )
24 {
25     if ( denominator == 0 )
26         throw DivideByZeroException();
27
28     return static_cast< double > ( numerator ) / denominator;
29 }

```

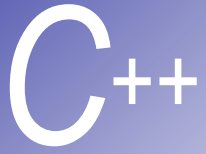
Eğer **denominator == 0** olursa Exception' u çıkaraçak bir fonksiyon tasarlanmış durumda

C++

```
30
31 // Driver program
32 int main()
33 {
34     int number1, number2;
35     double result;
36
37     cout << "Enter two integers (end-of-file to end): ";
38
39     while ( cin >> number1 >> number2 ) {
40
41         // the try block wraps the code that may throw an
42         // exception and the code that should not execute
43         // if an exception occurs
44         try {
45             result = quotient( number1, number2 );
46             cout << "The quotient is: " << result << endl;
47         }
48         catch ( DivideByZeroException ex ) { // exception handler
49             cout << "Exception occurred: " << ex.what() << '\n';
50         }
51
52         cout << "\nEnter two integers (end-of-file to end): ";
53     }
54
55     cout << endl;
56     return 0;        // terminate normally
57 }
```

try bloğu, exception olduğunda kodun çalıştırılmaması için

catch bloğu **try** bloğunu takip ediyor, ve exception-handling kodu içeriyor.



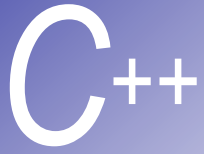
Program Çıktısı

```
Enter two integers (end-of-file to end): 100 7  
The quotient is: 14.2857
```

```
Enter two integers (end-of-file to end): 100 0  
Exception occurred: attempted to divide by zero
```

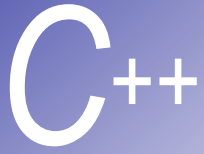
```
Enter two integers (end-of-file to end): 33 9  
The quotient is: 3.66667
```

```
Enter two integers (end-of-file to end):
```

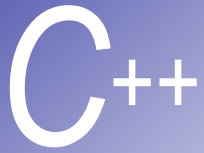
Dosya İşlemleri

- C++ programlarıyla veri dosyaları yaratılabilir, güncellenebilir ve işleme sokulabilir.
 - Dosyalar çok miktarda veriyi sürekli olarak depolamak amacıyla kullanılır.
 - Değişkenlerde ve dizilerde verilerin depolanması sadece geçicidir.



Veri Hiyerarşisi

- Bit – en küçük veri ögesi
 - 0 veya 1 değerli
- Byte – 8 bit
 - Karakter kaydetmek için kullanılır
 - Ondalıklı sayılar, harfler ve özel semboller
- Field (Alan) - Anlamlı kelime grupları
 - Örnek: Adınız

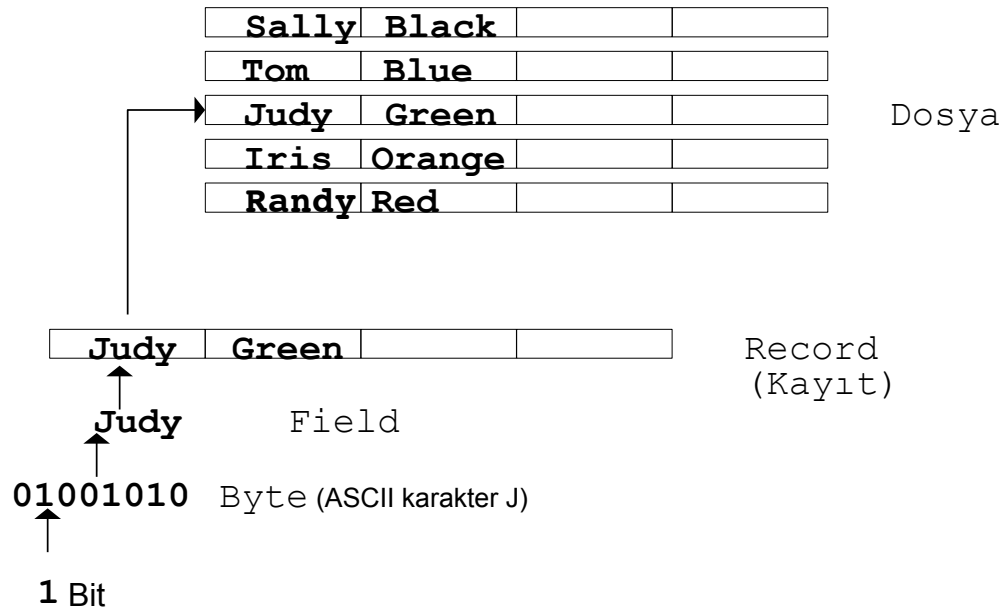


Veri Hiyerarşisi

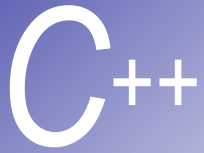
- Record (Kayıt) – İlgili alanlar topluluğu
 - **struct** veya **class** ile temsil edilir
 - Örnek: Maaş bordrosu sisteminde, her bir grup çalışan için kimlik numarası, ad, adres, vs. içeren bir kayıt tutulması
- Dosya – İlgili record topluluğu
 - Örnek: Maaş bordrosu dosyası
- Veritabanı – İlgili dosyalar topluluğu

C++

Veri Hiyerarşisi

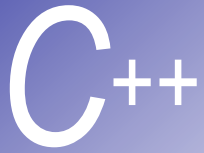


- **Kayıt Anahtarı (Record key)**
 - Dosyadaki özel bir kayıtnın geri çağrılışını kolaylaştıran bir kayıttır.
- **Sıralı Dosya (Sequential file)**
 - Kayıt anahtarına göre sıralanmış kayıtlar



Dosyalar ve Streams

- C++ her dosyayı byte'ların sıralanışı olarak görür
 - Dosyalar *end-of-file işaretçisi* ile sonlanır
- Stream bir dosya açıldığında oluşur
- Dosya işlemleri
 - Başlıklar `<iostream.h>` ve `<fstream.h>`
 - `class ifstream` - giriş
 - `class ofstream` - çıkış
 - `class fstream` – giriş veya çıkış



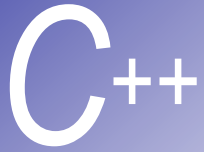
Sıralı Erişimli Dosya Oluşturma

- Dosyalar **`ifstream`**, **`ofstream`** veya **`fstream`** stream sınıflarının nesneleri oluşturulurken açılır
- **`file`** nesnesi için dosya stream elemanı fonksiyonları :
 - **`file.open("Filename", fileOpenMode) ;`**
 - **`file.close() ;`**
 - Dosya açık bir şekilde kapatılmadıysa destructor otomatik olarak dosyayı kapatır.

Nesne ve dosya ile
haberleşme satırı

"line of communication"
oluşturur





Dosya açma durumları:

Durum	Açıklama
ios::app	Bütün çıktıları dosyanın sonuna yaz.
ios::ate	Dosyayı çıkış için aç, dosyanın sonuna git (genelde dosyaya birşeyler eklemek için kullanılır). Veri dosyanın herhangi bir yerine yazılabilir.
ios::in	Dosyayı giriş için aç.
ios::out	Dosyayı çıkış için aç.
ios::trunc	Dosya eğer varsa içeriğini siler (ios::out için varsayılan durumla aynı)
ios::binary	Dosyayı binary (text olmayan) giriş/çıkış için aç

C++

Sıralı Erişimli Dosya Oluşturma

```
1 // Fig. 14.4: fig14_04.cpp
2 // Create a sequential file
3 #include <iostream>
4
5 using std::cout;
6 using std::cin;
7 using std::ios;
8 using std::cerr;
9 using std::endl;
10
11 #include <fstream>
12
13 using std::ofstream;
14
15 #include <cstdlib>
16
17 int main()
18 {
19     // ofstream constructor opens file
20     ofstream outClientFile( "clients.dat", ios::out );
```

ofstream nesneleri dosyayı çıkış (dosyaya yazma) için açar. Eğer "**clients.dat**" dosyası yoksa oluşturulur. **ios::out**, **ofstream** nesneleri için varsayılan açılış durumudur.

C++

Sıralı Erişimli Dosya Oluşturma

```
21
22  if ( !outClientFile ) { // overloaded ! operator
23      cerr << "File could not be opened" << endl;
24      exit( 1 ); // prototype in cstdlib
25  }
26
27  cout << "Enter the account, name, and balance.\n"
28        << "Enter end-of-file to end input.\n? ";
29
30  int account;
31  char name[ 30 ];
32  double balance;
33
34  while ( cin >> account >> name >> balance ) {
35      outClientFile << account << ' ' << name
36                  << ' ' << balance << '\n';
```

Aşırı yüklenmiş **operator!**, eğer **failbit** veya **badbit** ayarlı ise **true** dönderir.

Cout ile çıktı işlemleri yapılacağına **clients.dat** dosyasına bağlantı yapılmış **outClientFile**'a çıktı işlemleri yapılıyor.

Girdi verileri. end-of-file veya kötü veri girilirse, **cin 0** dönderir (normalde **cin 1** dönderir), ve **while** döngüsü sonlanır.

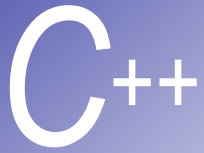
C++

Sıralı Erişimli Dosya Oluşturma

```
37         cout << "? ";  
38     }  
39  
40     return 0; // ofstream destructor closes file  
41 }
```

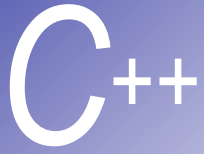
```
Enter the account, name, and balance.  
Enter EOF to end input.  
? 100 Jones 24.98  
? 200 Doe 345.67  
? 300 White 0.00  
? 400 Stone -42.16  
? 500 Rich 224.62  
? ^Z
```

outClientFile'in bozucusu
otomatik olarak **client.dat**
dosyasını kapatıyor



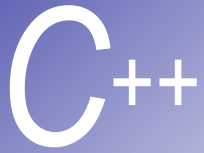
Sıralı Erişimli Dosyadan Veri Okuma

- Dosya yer imleçleri için dosya streamleri üye fonksiyonları:
 - istream için seekg (seek get) ve ostream için seekp (seek put)
 - // fileObject' in n' inci byte' tına hareket ettirir
 - // ios::beg farz edilir
fileObject.seekg(n);
 - // fileObject' te n byte ileri hareket ettirir
fileObject.seekg(n, ios::cur);



Sıralı Erişimli Dosyadan Veri Okuma

- Dosya yer imleçleri için dosya streamleri üye fonksiyonları:
 - // fileObject' in sonundan y byte geri hareket ettirir
`fileObject.seekg(y, ios::end);`
 - // fileObject' in sonuna (EOF) hareket ettirir
`fileObject.seekg(0, ios::end);`
- `tellg` ve `tellp` – imlecin o zamanda bulunduğu konumu dönderir
- `location = fileObject.tellg()` //long dönderir



Sıralı dosyadan okuma

```
1 // Fig. 14.7: fig14_07.cpp
2 // Reading and printing a sequential file
3 #include <iostream>
4
5 using std::cout;
6 using std::cin;
7 using std::ios;
8 using std::cerr;
9 using std::endl;
10
11 #include <fstream>
12
13 using std::ifstream;
14
15 #include <iomanip>
16
17 using std::setiosflags;
18 using std::resetiosflags;
19 using std::setw;
20 using std::setprecision;
```

C++

Sıralı dosyadan okuma

```
21
22 #include <cstdlib>
23
24 void outputLine( int, const char * const, double );
25
26 int main()
27 {
28     // ifstream constructor opens the file
29     ifstream inClientFile( "clients.dat", ios::in );
30
31     if ( !inClientFile ) {
32         cerr << "File could not be opened\n";
33         exit( 1 );
34     }
35
36     int account;
37     char name[ 30 ];
38     double balance;
39
```

"clients.dat"
dosyasını giriş
(dosyadan veri alma)
için açma

C++

Sıralı dosyadan okuma

```

40     cout << setiosflags( ios::left ) << setw( 10 ) << "Account"
41         << setw( 13 ) << "Name" << "Balance\n"
42         << setiosflags( ios::fixed | ios::showpoint );
43
44     while ( inClientFile >> account >> name >> balance )
45         outputLine( account, name, balance );
46
47     return 0; // ifstream
48 }
49
50 void outputLine( int acct, const char * const name, double bal )
51 {
52     cout << setiosflags( ios::left ) << setw( 10 ) << acct
53         << setw( 13 ) << name << setw( 7 ) << setprecision( 2 )
54         << resetiosflags( ios::left )
55         << bal << '\n'; }

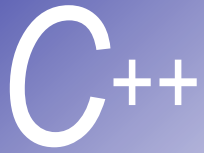
```

Account Name Balance

Veriyi okur, ve **account**, **name** ve **balance** olarak kaydeder. Datayı ekranda biçimlendirerek yazan **outputLine** fonksiyonu kullanılıyor.

Account	Name	Balance
100	Jones	24.98
200	Doe	345.67
300	White	0.00
400	Stone	-42.16
500	Rich	224.62

while döngüsü dosya sonuna kadar devam ediyor , 0 dönünce (**inClientFile** yerine) Destructor otomatik olarak **clients.dat** dosyasını kapatıyor.



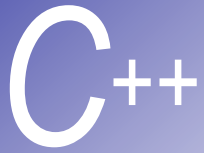
Sıralı dosyadan okuma

```
1 // Fig. 14.8: fig14_08.cpp
2 // Credit inquiry program
3 #include <iostream>
4
5 using std::cout;
6 using std::cin;
7 using std::ios;
8 using std::cerr;
9 using std::endl;
10
11 #include <fstream>
12
13 using std::ifstream;
14
15 #include <iomanip>
16
17 using std::setiosflags;
18 using std::resetiosflags;
19 using std::setw;
20 using std::setprecision;
```


C++

Sıralı dosyadan okuma

```
21
22 #include <cstdlib>
23
24 enum RequestType { ZERO_BALANCE = 1, CREDIT_BALANCE,
25                   DEBIT_BALANCE, END };
26 int getRequest();
27 bool shouldDisplay( int, double );
28 void outputLine( int, const char * const, double );
29
30 int main()
31 {
32     // ifstream constructor opens the file
33     ifstream inClientFile( "clients.dat", ios::in );
34
35     if ( !inClientFile ) {
36         cerr << "File could not be opened" << endl;
37         exit( 1 );
38     }
39
```



Sıralı dosyadan okuma

```
41     char name[ 30 ];
42     double balance;
43
44     cout << "Enter request\n"
45           << " 1 - List accounts with zero balances\n"
46           << " 2 - List accounts with credit balances\n"
47           << " 3 - List accounts with debit balances\n"
48           << " 4 - End of run"
49           << setiosflags( ios::fixed | ios::showpoint );
50     request = getRequest();
51
52     while ( request != END ) {
53
54         switch ( request ) {
55             case ZERO BALANCE:
56                 cout << "\nAccounts with zero balances:\n";
57                 break;
58             case CREDIT BALANCE:
59                 cout << "\nAccounts with credit balances:\n";
60                 break;
61             case DEBIT BALANCE:
62                 cout << "\nAccounts with debit balances:\n";
63                 break;
64         }
```

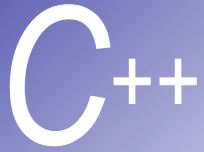
C++

Sıralı dosyadan okuma

```
65
66     inClientFile >> account >> name >> balance;
67
68     while ( !inClientFile.eof() ) {
69         if ( shouldDisplay( request, balance ) )
70             outputLine( account, name, balance );
71
72         inClientFile >> account >> name >> balance;
73     }
74
75     inClientFile.clear();      // reset eof for next input
76     inClientFile.seekg( 0 );  // move to beginning of file
77     request = getRequest();
78 }
79
80 cout << "End of run." << endl;
81
82 return 0;    // ifstream destructor closes the file
83 }
84
```

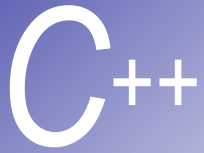
Verileri dosyadan oku,
account, **name** ve
balance'e kopyala

Dosya imlecini
dosyanın başına koy.



Sıralı dosyadan okuma

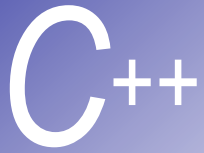
```
85 int getRequest()
86 {
87     int request;
88
89     do {
90         cout << "\n? ";
91         cin >> request;
92     } while( request < ZERO BALANCE && request > END );
93
94     return request;
95 }
96
97 bool shouldDisplay( int type, double balance )
98 {
99     if ( type == CREDIT BALANCE && balance < 0 )
100         return true;
101
102     if ( type == DEBIT BALANCE && balance > 0 )
103         return true;
104
105     if ( type == ZERO BALANCE && balance == 0 )
106         return true;
107
108     return false;
109 }
```



Sıralı dosyadan okuma

```
110
111 void outputLine( int acct, const char * const name, double bal )
112 {
113     cout << setiosflags( ios::left ) << setw( 10 ) << acct
114         << setw( 13 ) << name << setw( 7 ) << setprecision( 2 )
115         << resetiosflags( ios::left )
116         << bal << '\n';
117 }
```

```
Enter request
 1 - List accounts with zero balances
 2 - List accounts with credit balances
 3 - List accounts with debit balances
 4 - End of run
? 1
Accounts with zero balances:
300      White           0.00
? 2
Accounts with credit balances:
400      Stone          -42.16
? 3
Accounts with debit balances:
100      Jones           24.98
200      Doe             345.67
500      Rich            224.62
? 4
End of run.
```



Sıralı Erişimli Dosyaları Güncelleme

- Sıralı erişimli dosya
 - Diğer verilere zarar vermeden dosyayı değiştirme riski olmaksızın değişiklik yapılamaz

300 White 0.00 400 Jones 32.87 (dosyadaki eski veri)

White'in ismini Worthington olarak değiştirmek istersek ,

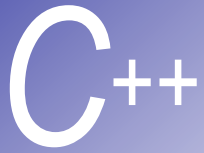
300 Worthington 0.00



300 White 0.00 400 Jones 32.87 → 300 Worthington 0.00ones 32.87

Verinin üzerine yazıldı

- Dosyaya çıktı şeklinde olan formatlı metin iç gösterimden çok farklıdır



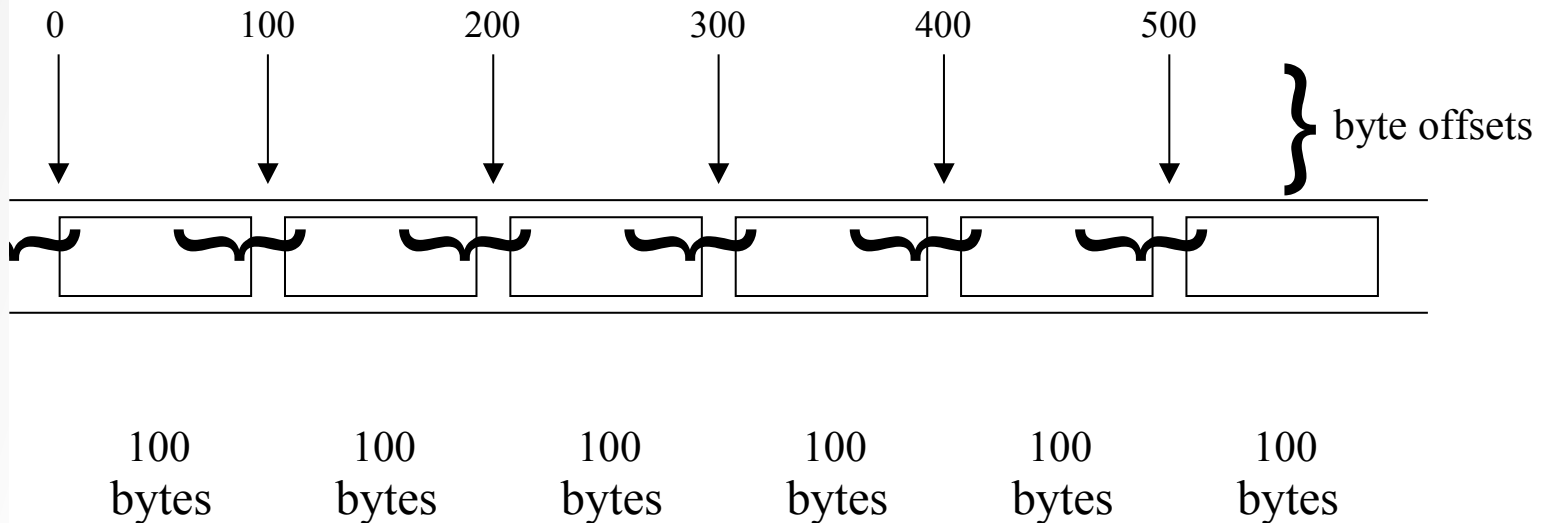
Rastgele (Doğrudan) Erişimli Dosyalar

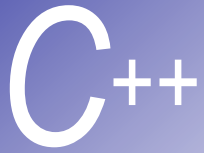
- Rastgele erişimli dosya
 - Diğer kayıtlı şeyleri araştırmadan doğrudan istenilen veri veya verilere ulaşma
 - Dosyadaki kayıtlara anında ulaşım
 - Diğer verilere zarar vermeden veriyi araya yerleştirme
 - Daha önceden girilmiş veriyi üstüne yazmadan güncelleme ya da silme.

C++

Rastgele (Doğrudan) Erişimli Dosyalar

- Sabit uzunluktaki kayıtlar kullanarak yapılır:

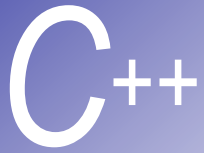




Rastgele (Doğrudan) Erişimli Dosya Oluşturma

- **write** – bellekte özel bir konumdan başlayarak sabit sayıda byte belirtilen stream' e çıktı olarak verilir
 - tamsayı değerlikli **number** dosyaya yazılırken ,

```
outFile.write( reinterpret_cast<const char *>( &number ),  
sizeof( number ) );
```
 - Birinci argüman: **const char *** türünden pointer (nerden yazılacağı)
 - **number**' in adresi bir pointer' a cast ediliyor



Rastgele (Doğrudan) Erişimli Dosya Oluşturma

- **write** (devam)
 - İkinci argüman: yazılacak byte sayısı (**sizeof (number))**
 - Verilerin dahilen binary şekilde temsil edildiğini unutmayınız (bu yüzden tamsayılar 4 byte' da depolanabilir)
 - Aşağıdaki yapıyı kullanma
outFile << number;
 - 1' den 11 basamağa kadar yazabilir, her bir basamak bir byte'lık yer kaplar

C++

Rastgele (Doğrudan) Erişimli Dosyalar

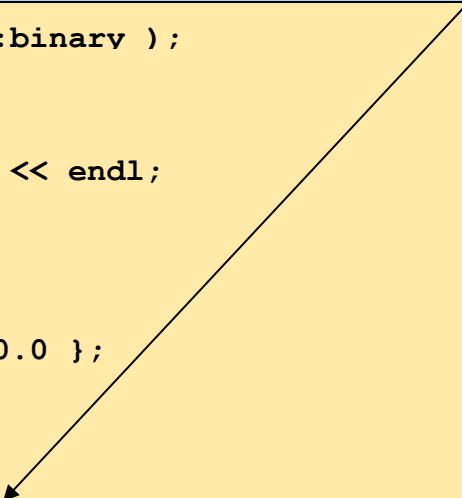
```
1 // Fig. 14.11: clntdata.h
2 // Definition of struct clientData used in
3 // Figs. 14.11, 14.12, 14.14 and 14.15.
4 #ifndef CLNTDATA H
5 #define CLNTDATA H
6
7 struct clientData {
8     int accountNumber;
9     char lastName[ 15 ];
10    char firstName[ 10 ];
11    double balance;
12 };
13
14 #endif
15 // Fig. 14.11: fig14_11.cpp
16 // Creating a randomly accessed file sequentially
17 #include <iostream>
18
19 using std::cerr;
20 using std::endl;
21 using std::ios;
22
23 #include <fstream>
```

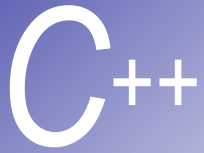
C++

Rastgele (Doğrudan) Erişimli Dosyalar

```
24
25 using std::ofstream;
26
27 #include <cstdlib>
28
29 #include "clntdata.h"
30
31 int main()
32 {
33     ofstream outCredit( "credit.dat", ios::binary );
34
35     if ( !outCredit ) {
36         cerr << "File could not be opened." << endl;
37         exit( 1 );
38     }
39
40     clientData blankClient = { 0, "", "", 0.0 };
41
42     for ( int i = 0; i < 100; i++ )
43         outCredit.write(
44             reinterpret cast<const char *>( &blankClient ),
45             sizeof( clientData ) );
46     return 0;
47 }
```

Aynı boyutlu veri gruplarının dosyaya yazılması ile rastgele erişimli dosya oluşturuluyor. Her grup **sizeof(clientData)** boyutunda





Doğrudan Erişimli Dosyaya Rastgele Yazma

- **seekp**, **write** ile birlikte kullanılarak çıktı dosyasında spesifik bir yere veri kaydedilebilir.

```
3  #include <iostream>
4
5  using std::cerr;
6  using std::endl;
7  using std::cout;
8  using std::cin;
9  using std::ios;
10
11 #include <fstream>
12
13 using std::ofstream;
14
15 #include <cstdlib>
16 #include "clntdata.h"
```

C++

Doğrudan Erişimli Dosyaya Rastgele Yazma

```
17
18 int main()
19 {
20     ofstream outCredit( "credit.dat", ios::binary );
21
22     if ( !outCredit ) {
23         cerr << "File could not be opened." << endl;
24         exit( 1 );
25     }
26
27     cout << "Enter account number "
28           << "(1 to 100, 0 to end input)\n? ";
29
30     clientData client;
31     cin >> client.accountNumber;
32
```

Enter account number (1 to 100, 0 to end input)

C++

Doğrudan Erişimli Dosyaya Rastgele Yazma

```
33 while ( client.accountNumber > 0 &&
34         client.accountNumber <= 100 ) {
35     cout << "Enter lastname, firstname, balance\n? ";
36     cin >> client.lastName >> client.firstName
37         >> client.balance;
38
39     outCredit.seekp( ( client.accountNumber - 1 ) *
40                     sizeof( clientData ) );
41     outCredit.write(
42         reinterpret_cast<const char *>( &client ),
43         sizeof( clientData ) );
44
45     cout << "Enter account number\n? ";
46     cin >> client.accountNumber;
47 }
48
49 return 0;
50 }
```

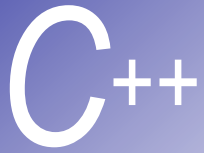
Enter lastname, firstname, balance
? Barker Doug 0.00

Dosyada uygun yeri bul. Gruplar **sizeof(clientData)** boyutunda, ve ilk pozisyon 0' da. Hesaplar **accountNumber** sırasına göre dizililer

C++

Doğrudan Erişimli Dosyaya Rastgele Yazma

```
Enter account number (1 to 100, 0 to end input)
? 37
Enter lastname, firstname, balance
? Barker Doug 0.00
Enter account number
? 29
Enter lastname, firstname, balance
? Brown Nancy -24.54
Enter account number
? 96
Enter lastname, firstname, balance
? Stone Sam 34.98
Enter account number
? 88
Enter lastname, firstname, balance
? Smith Dave 258.34
Enter account number
? 33
Enter lastname, firstname, balance
? Dunn Stacey 314.33
Enter account number
? 0
```

Dosyadan Sıralı Şekilde Veri Okuma

- **read**
 - belirli byte'lık verinin belirtilen stream'den bellekte belirtilen adresten başlayan alana girişini yapar
 - **Write** ile benzer yapıdadır

```
inFile.read( reinterpret_cast<char*>( &number ), sizeof( int ) );
```
 - Birinci argüman: **char *** türünden pointer (byte'ları koymak için yer)
 - İkinci argüman: alınacak byte sayısı :

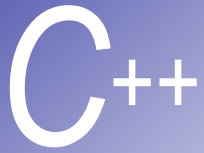
```
sizeof( int )
```
 - Şunu kullanma

```
inFile >> number;
```
- doğrudan erişim teknikleriyle hızlı sıralama

C++

Dosyadan Sıralı Şekilde Veri Okuma

```
1 // Fig. 14.14: fig14_14.cpp
2 // Reading a random access file sequentially
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7 using std::ios;
8 using std::cerr;
9
10 #include <iomanip>
11
12 using std::setprecision;
13 using std::setiosflags;
14 using std::resetiosflags;
15 using std::setw;
16
17 #include <fstream>
18
19 using std::ifstream;
20 using std::ofstream;
21
```



Dosyadan Sıralı Şekilde Veri Okuma

```
22 #include <cstdlib>
23 #include "clntdata.h"
24
25 void outputLine( ostream&, const clientData & );
26
27 int main()
28 {
29     ifstream inCredit( "credit.dat", ios::in );
30
31     if ( !inCredit ) {
32         cerr << "File could not be opened." << endl;
33         exit( 1 );
34     }
35
36     cout << setiosflags( ios::left ) << setw( 10 ) << "Account"
37          << setw( 16 ) << "Last Name" << setw( 11 )
38          << "First Name" << resetiosflags( ios::left )
39          << setw( 10 ) << "Balance" << endl;
40
41     clientData client;
```

C++

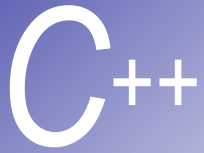
Dosyadan Sıralı Şekilde Veri Okuma

```
43   inCredit.read( reinterpret_cast<char *>( &client ),
44                 sizeof( clientData ) );
45
46   while ( inCredit && !inCredit.eof() ) {
47
48       if ( client.accountNumber != 0 )
49           outputLine( cout, client );
50
51       inCredit.read( reinterpret_cast<char *>( &client ),
52                     sizeof( clientData ) );
53   }
54
55   return 0;
56 }
57
```

sizeof(clientData) bytelık veriyi alır ve **&client**'a yazar (**client**'ın bellek adresine).

eof fonksiyonu kullanılarak dosyanın sonuna gelinip gelinmediği kontrol ediliyor

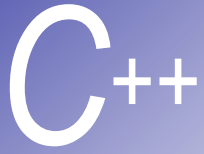
Client'taki verileri ekrana yazdırır



Dosyadan Sıralı Şekilde Veri Okuma

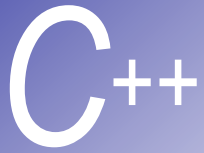
```
58 void outputLine( ostream &output, const clientData &c )
59 {
60     output << setiosflags( ios::left ) << setw( 10 )
61         << c.accountNumber << setw( 16 ) << c.lastName
62         << setw( 11 ) << c.firstName << setw( 10 )
63         << setprecision( 2 ) << resetiosflags( ios::left )
64         << setiosflags( ios::fixed | ios::showpoint )
65         << c.balance << '\n';
```

Account	Last Name	First Name	Balance
29	Brown	Nancy	-24.54
33	Dunn	Stacey	314.33
37	Barker	Doug	0.00
88	Smith	Dave	258.34
96	Stone	Sam	34.98



Örnek: Transaction İşlemi

- Aşağıdaki örnek rastgele erişimli dosyaları kullanarak banka hesap bilgilerine anında erişmeye yöneliktir.
- Aşağıdaki işlemler yapılacaktır
 - Varolan hesapları güncelleme
 - Yeni hesaplar ekleme
 - Hesapları silme
 - Bütün hesapların formatlı bir biçimde listesi bir text dosyasına yazılması



Örnek: Transaction İşlemi

```
1 // Fig. 14.15: fig14_15.cpp
2 // This program reads a random access file sequentially,
3 // updates data already written to the file, creates new
4 // data to be placed in the file, and deletes data
5 // already in the file.
6 #include <iostream>
7
8 using std::cout;
9 using std::cerr;
10 using std::cin;
11 using std::endl;
12 using std::ios;
13
14 #include <fstream>
15
16 using std::ofstream;
17 using std::ostream;
18 using std::fstream;
19
20 #include <iomanip>
```

C++

Örnek: Transaction İşlemi

```
21
22 using std::setiosflags;
23 using std::resetiosflags;
24 using std::setw;
25 using std::setprecision;
26
27 #include <cstdlib>
28 #include "clntdata.h"
29
30 int enterChoice();
31 void textFile( fstream& );
32 void updateRecord( fstream& );
33 void newRecord( fstream& );
34 void deleteRecord( fstream& );
35 void outputLine( ostream&, const clientData & );
36 int getAccount( const char * const );
37
38 enum Choices { TEXTFILE = 1, UPDATE, NEW, DELETE, END };
39
```


C++

Örnek: Transaction İşlemi

```
40 int main()
41 {
42     fstream inOutCredit( "credit.dat", ios::in | ios::out );
43
44     if ( !inOutCredit ) {
45         cerr << "File could not be opened." << endl;
46         exit ( 1 );
47     }
48
49     int choice;
50
51     while ( ( choice = enterChoice() ) != END ) {
52
53         switch ( choice ) {
54             case TEXTFILE:
55                 textFile( inOutCredit );
56                 break;
57             case UPDATE:
58                 updateRecord( inOutCredit );
59                 break;
```

fstream nesneleri hem giriş hem de çıkış için kullanılır

enterChoice
fonksiyonunu çağırır

C++

Örnek: Transaction İşlemi

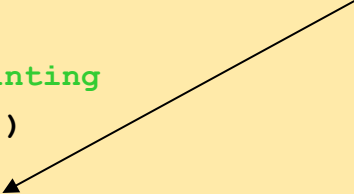
```
60         case NEW:
61             newRecord( inOutCredit );
62             break;
63         case DELETE:
64             deleteRecord( inOutCredit );
65             break;
66         default:
67             cerr << "Incorrect choice\n";
68             break;
69     }
70
71     inOutCredit.clear(); // resets end-of-file indicator
72 }
73
74 return 0;
75 }
76
77 // Prompt for and input menu choice
78 int enterChoice()
79 {
```

C++

Örnek: Transaction İşlemi

```
80     cout << "\nEnter your choice" << endl
81         << "1 - store a formatted text file of accounts\n"
82         << "    called \"print.txt\" for printing\n"
83         << "2 - update an account\n"
84         << "3 - add a new account\n"
85         << "4 - delete an account\n"
86         << "5 - end program\n? ";
87
88     int menuChoice;
89     cin >> menuChoice;
90     return menuChoice;
91 }
92
93 // Create formatted text file for printing
94 void textFile( fstream &readFromFile )
95 {
96     ofstream outPrintFile( "print.txt", ios::out );
97
98     if ( !outPrintFile ) {
99         cerr << "File could not be opened." << endl;
100        exit( 1 );
```

print.txt dosyasını açar veya oluşturur



C++

Örnek: Transaction İşlemi

```
101     }
102
103     outPrintFile << setiosflags( ios::left ) << setw( 10 )
104         << "Account" << setw( 16 ) << "Last Name" << setw( 11 )
105         << "First Name" << resetiosflags( ios::left )
106         << setw( 10 ) << "Balance" << endl;
107     readFromFile.seekg( 0 );
108
109     clientData client;
110     readFromFile.read( reinterpret_cast<char *>( &client ),
111         sizeof( clientData ) );
112
113     while ( !readFromFile.eof() ) {
114         if ( client.accountNumber != 0 )
115             outputLine( outPrintFile, client );
116
117         readFromFile.read( reinterpret_cast<char *>( &client ),
118             sizeof( clientData ) );
119     }
120 }
```


*formatlı veriyi
print.txt
dosyasına yazar*

C++


Örnek: Transaction İşlemi

```
121
122// Update an account's balance
123void updateRecord( fstream &updateFile )
124{
125    int account = getAccount( "Enter account to update" );
126
127    updateFile.seekg( ( account - 1 ) * sizeof( clientData ) );
128
129    clientData client;
130    updateFile.read( reinterpret_cast<char *>( &client ),
131                    sizeof( clientData ) );
132
133    if ( client.accountNumber != 0 ) {
134        outputLine( cout, client );
135        cout << "\nEnter charge (+) or payment (-): ";
136
137        double transaction;    // charge or payment
138        cin >> transaction;    // should validate
139        client.balance += transaction;
140        outputLine( cout, client );
```

hesap numarasına göre dosyada
belirli bir yere hareket edilir



Client' a veri girişi



C++

Örnek: Transaction İşlemi

```
141     updateFile.seekp( ( account-1 ) * sizeof( clientData ) );
142     updateFile.write(
143         reinterpret_cast<const char *>( &client ),
144         sizeof( clientData ) );
145 }
146 else
147     cerr << "Account #" << account
148         << " has no information." << endl;
149 }
150
151 // Create and insert new record
152 void newRecord( fstream &insertInFile )
153 {
154     int account = getAccount( "Enter new account number" );
155
156     insertInFile.seekg( ( account-1 ) * sizeof( clientData ) );
157
158     clientData client;
159     insertInFile.read( reinterpret_cast<char *>( &client ),
160                     sizeof( clientData ) );
161 }
```

Hesap bakiyesini
değiştir, dosyada
belirli yeri bul ve
dosyaya yaz.

C++

Örnek: Transaction İşlemi

```

162  if ( client.accountNumber == 0 ) {
163      cout << "Enter lastname, firstname, balance\n? ";
164      cin >> client.lastName >> client.firstName
165          >> client.balance;
166      client.accountNumber = account;
167      insertInFile.seekp( ( account - 1 ) *
168                          sizeof( clientData ) );
169      insertInFile.write(
170          reinterpret cast<const char *>( &client ),
171          sizeof( clientData ) );
172  }
173  else
174      cerr << "Account #" << account
175          << " already contains information." << endl;
176 }
177
178 // Delete an existing record
179 void deleteRecord( fstream &deleteFromFile )
180 {
181     int account = getAccount( "Enter account to delete" );
182

```

Yeni kayıt oluşturur.

Hesabın varlığını
kontrol et.

Veri gir.

İmlecin konumu

Yeni verileri dosyaya yaz

C++

Örnek: Transaction İşlemi

```
184
185     clientData client;
186     deleteFromFile.read( reinterpret cast<char *>( &client ),
187                         sizeof( clientData ) );
188
189     if ( client.accountNumber != 0 ) {
190         clientData blankClient = { 0, "", "", 0.0 };
191
192         deleteFromFile.seekp( ( account - 1 ) *
193                             sizeof( clientData ) );
194         deleteFromFile.write(
195             reinterpret cast<const char *>( &blankClient ),
196             sizeof( clientData ) );
197         cout << "Account #" << account << " deleted." << endl;
198     }
199     else
200         cerr << "Account #" << account << " is empty." << endl;
201 }
202
203 // Output a line of client information
204 void outputLine( ostream &output, const clientData &c )
205 {
```

Boş hesapla yer
değiştirerek
kayıdı sil.

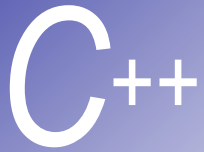
C++

Örnek: Transaction İşlemi

```
206     output << setiosflags( ios::left ) << setw( 10 )
207           << c.accountNumber << setw( 16 ) << c.lastName
208           << setw( 11 ) << c.firstName << setw( 10 )
209           << setprecision( 2 ) << resetiosflags( ios::left )
210           << setiosflags( ios::fixed | ios::showpoint )
211           << c.balance << '\n';
212 }
213
214 // Get an account number from the keyboard
215 int getAccount( const char * const prompt )
216 {
217     int account;
218
219     do {
220         cout << prompt << " (1 - 100): ";
221         cin >> account;
222     } while ( account < 1 || account > 100 );
223
224     return account;
225 }
```



Formatlı çıktı.



Örnek: Transaction İşlemi

AFTER OPTION 1 PRINT.TXT CONTAINS:

Account	Last Name	First Name	Balance
29	Brown	Nancy	-24.54
33	Dunn	Stacey	314.33
37	Barker	Doug	0.00
88	Smith	Dave	258.34
96	Stone	Sam	34.98

Enter account to update (1 - 100): 37

37	Barker	Doug	0.00
----	--------	------	------

Enter charge (+) or payment (-): +87.99

37	Barker	Doug	87.99
----	--------	------	-------

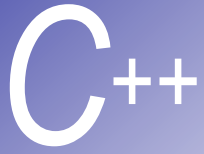
Enter new account number (1 - 100): 22

Enter lastname, firstname, balance

? Johnston Sarah 247.45

Enter account to delete (1 - 100): 29

Account #29 deleted.



Nesnelerin Giriş / Çıkışları

- Nesne veri üyeleri bir disk dosyasına çıktı olduğu zaman
 - nesnenin tür bilgileri kaybedilir
 - sadece veri byte'ları bilinir
- Olası çözüm:
 - nesnenin türünü belirterek nesne çıkışına devam et