Bölüm 21 - C++ Stream Input/Output

İçerik

Bölüm 21 - C++ Stream Input/Output

İçerik (devam)

21.1 Giriş

- Çoğu C++ I/O özellikleri nesne tabanlıdır
  - Referansları, fonksiyon ve operatör yüklemeleri kullan
- C++ tip korumalı I/O kullanır
  - Her I/O operasyonu veri tipine duyarlı olacak şekilde gerçekleşir
- Genişlik
  - Kullanıcı standart tiplerin yanısıra kullanıcı tanımlı I/O tipi belirleyebilir

21.2 Akış(Stream)lar

- Stream (Akış)
  - Bir Byte-lar dizisi formunda bilgi transferi

- I/O Operasyonları:
  - Input: Bir girdi aygıtından (klavye, disk sürücü, network bağlantısı gibi) ana belleğe geçen bir stream
  - Output: Ana bellekten bir çıktı aygıtına (ekran, yazıcı, disk sürücü, network bağlantısı gibi) geçen bir stream

21.2 Akış(Stream)lar

- I/O operasyonları bir darboğazdır
  - Bir stream-in akış zamanı, stream-deki verinin CPU tarafından işlenme zamanından çok daha fazladır
- Alt-düzey I/O
  - Formatsız
  - İlgiye göre keyfi byte birimi
  - Yüksek hız, yüksek hacim fakat uygun/kullanışlı değil
- Yüksek-düzey I/O
  - Formatlı
  - Byte-lar anlamlı birimlere bölünür : tamsayılar, karakterler v.s.
  - Yüksek hacimli dosya işleme hariç tüm I/O için iyi.

21.2.1 `iostream` Library Header Dosyaları

- **`iostream`** kütüphanesi:
  - **`<iostream.h>`: `cin`, `cout`, `cerr`,** ve **`clog`** nesnelerini içerir
  - **`<iomanip.h>`:** *parameteleştirilmiş stream düzenleyicilerini* içerir
  - **`<fstream.h>`:** kullanıcı kontrollü dosya işleme operasyonlarında önemli olan bilgileri içerir

21.2.2 Stream Input/Output Sınıfları ve Nesneleri

- **`ios`:**
  - **`istream`** ve **`ostream`** **`ios`** dan kalıttır
    - **`iostream`**, **`istream`** ve **`ostream`** den kalıttır.
- **`<<`** (sol-kaydırma operatörü)
  - *stream yerleştirme (insertion) operatörü* olarak yüklenir
- **`>>`** (sağ-kaydırma operatörü)
  - *stream çıkarma (extraction) operatörü* olarak yüklenir
  - Her iki operatör de **`cin`**, **`cout`**, **`cerr`**, **`clog`**, ve kullanıcı-tanımlı stream nesneleri ile kullanılır

21.2.2 Stream Input/Output Sınıfları ve Nesneleri

- **`istream:`** input stream

  `cin >> birDegisken;`

  - **`cin birDegisken`** e ne tip bir veri atanacağını bilir (**`birDegisken`** nin tipine göre).

- **`ostream:`** output stream
  - **`cout << birDegisken;`**
    - **`cout`** çıktı veri tipini bilir
  - **`cerr << birString;`**
    - **`birString`** i hemen yaz.
  - **`clog << birString;`**
    - **`birString`** i çıktı desteği dolduktan sonra yaz

21.3 Stream Output

- **`ostream`**: formatlı/formatsız çıktı uygular
  - Karakterler için **`put`** ve formatlanmamış karakterler için **`write`** kullanır
  - Sayı çıktıları: decimal, octal ve hexadecimal
  - Reel sayılar için değişik duyarlılık
  - Formatlı metin çıktısı

21.3.1 Stream-Insertion Operatörü

- **<<** hazır tipler için çıktı
  - Kullanıcı tipli çıktı için de kullanılır
  - **`cout << '\n';`**
    - Yenisatır karakterini yazar
  - **`cout << endl;`**
    - **`endl`** stream düzenleyici – yenisatır karakterini uygulayıp, çıktı desteğini boşaltır.
  - **`cout << flush;`**
    - **`flush`** çıktı desteğini boşaltır

21.3.2 Taşırma Stream-Insertion/Extraction Operatörleri

- **<<** : soldan sağa ilişkilendirme, sol operatör nesnesine (örneğin **cout**) referans gönderir
  - Çoklu kullanıma (taşırmaya) izin:

```
cout << "C ye " << " göre " << " daha kolay?";
```

Parantez kullanın

```
cout << "1 + 2 = " << (1 + 2); //doğru

cout << "1 + 2 = " << 1 + 2;  //yanlış
```

21.3.3 `char *` Değişkenleri Çıktısı

- **`<< char *`** tipinde bir string değişkenini çıktı verir

- Aynı stringin ilk karakterinin adresini çıktı vermek için, değişken tipini **`void *`** yap

21.3.3 `char *` Değişkenleri Çıktısı

```
1  // Fig. 21.8: fig21_08.cpp
2  // char* değişkenine yüklenen adresi yazma
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  int main()
9  {
10    char *string = "test";
11
12    cout << "Stringin değeri: " << string
13         << "\n static cast< void * >( string ) değeri: "
14         << static_cast< void * >( string ) << endl;
15    return 0;
16 }
```

```
Value of string is: test
static_cast< void *>( string ) değeri: 146a020
```

1stringi gir

2. Stringi yaz

2.1 `void * tipi ver`

2.2 pointer değerini (stringin adresini) yaz

21.3.4 `put` Stream Üye Fonksiyonu ile Karakter Çıktısı

- **`put`** üye fonksiyonu
  - Belirlenmiş akışa bir karakter çıktısı

    ```
    cout.put( 'A');
    ```
  - Çağrılan nesneye referans gönderir, birleştirilebilir

    ```
    cout.put( 'A' ).put( '\n' );
    ```
  - Bir ASCII-değer ifadesiyle çağrılabilir

    ```
    cout.put( 65 );
    ```
    - **`A`** çıktısını verir

- **>>** (stream-extraction)
  - Akış girdisi oluşturmak için kullanılır
  - Normalde whitespace (boşluk, tab, yenisatır) karakterlerini göz ardı eder
  - **EOF** gördüğünde sıfır(**false**) gönderir, aksi halde çağırdığı nesneye (örneğin **cin**) referans gönderir
    - Birleşik girdiye izin verir

```
cin >> x >> y;
```

- **>>** akışın bit durumunu kontrol eder
  - yanlış tip veri girişi ise **failbit** kurar
  - operasyon başarısız ise **badbit** kurar

21.4.1 Stream-Extraction Operatörü

- **>>** ve **<<** bağıl olarak yüksek önceliğe sahiptir
  - Koşullu ve aritmetik ifadeler parantez içinde olmalı

- Döngü kullanımının popüler yolu

```
while (cin >> not)
```

  - **EOF** görüldüğünde **0** (**false**) gönderir ve döngü sona erer

örü

```
1  // Fig. 21.11: fig21 11.cpp
2  // Stream-extraction operatörü - EOF DURUMUNDA SIFIR GÖNDERİR
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  int main()
10 {
11    int not, enbuyuk = -1;
12
13    cout << "Notu gir(Bitirmek için EOF gir): ";
14    while ( cin >> not ) {
15       if ( not > enbuyuk )
16          enbuyuk = not;
17
18       cout << "Notu gir(Bitirmek için EOF gir): ";
19    }
20
21    cout << "\n\nEn büyük not: " << enbuyuk << endl;
22    return 0;
23 }
```

1. Initialize variables

2. Perform loop

3. Output

Program Output

```
Notu gir(Bitirmek için EOF gir): 67
Notu gir(Bitirmek için EOF gir): 87
Notu gir(Bitirmek için EOF gir): 73
Notu gir(Bitirmek için EOF gir): 95
Notu gir(Bitirmek için EOF gir): 34
Notu gir(Bitirmek için EOF gir): 99
Notu gir(Bitirmek için EOF gir): ^Z
En büyük not: 99
```

21.4.2 `get` ve `getline` Üye Fonksiyonları

- **`cin.get()`**: akıştan bir karakter okur (whitespace de olabilir) ve gönderir

- **`cin.get( c )`**: akıştan bir karakter okur ve **`c`** ye aktarır

21.4.2 `get` ve `getline` Üye Fonksiyonları

- **`cin.get(dizi, boyut)`:**
  - 3 argument kabul eder: karakter dizisi, boyut limiti, ve ayırtaç (default **`'\n'`**).
  - Diziyi bir destek(tampon) gibi kullanır
  - Ayırtaç görüldüğünde, girdi akışında kalır
  - Diziye Null karakter eklenir
  - Ayırtaç akıştan kaldırılmadıkça, orada kalır

- **`cin.getline(dizi, boyut)`**
  - **`cin.get(destek, boyut)`** gibi işlem yapar fakat ayırtacı akıştan kaldırır , diziye yüklemez
  - Diziye Null karakter eklenir

21.4.2 get ve getline Üye Fonksiyonları

```
1  // Fig. 21.12: fig21_12.cpp
2  // Üye fonksiyonlar get, put  eof kullanımı.
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  int main()
10 {
11    char c;
12
13    cout << "Girdiden önce, cin.eof() değeri " << cin.eof()
14         << "\nEOF ile biten bir cümle yaz:\n";
15
16    while ( ( c = cin.get() ) != EOF )
17       cout.put( c );
18
19    cout << "\n Bu sistemdeki EOF : " << c;
20    cout << "\nGirdiden sonra, cin.eof() değeri " << cin.eof() << endl;
21    return 0;
22 }
```

```
Girdiden önce, cin.eof() değeri 0
EOF ile biten bir cümle yaz:
get ve put üye fonksiyonlarının testi^Z
get ve put üye fonksiyonlarının testi
Bu sistemdeki EOF: -1
Girdiden sonra, cin.eof() is 1
```

1. Initialize variables

2. Input data

2.1 Function call

3. Output

Program Output

```
1  // Fig. 21.14: fig21_14.cpp
2  // getline üye fonksiyonu ile karakter girdisi.
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  int main()
10 {
11    const int BOYUT = 80;
12    char destek[ BOYUT ];
13
14    cout << "Bir cümle gir:\n";
15    cin.getline( destek, BOYUT);
16
17    cout << "\nGirilen cümle:\n" << destek << endl;
18    return 0;
19 }
```

```
Bir cümle gir:
getline üye fonksiyonu kullanımı

Girilen cümle:
getline üye fonksiyonu kullanımı
```

1. Initialize variables

2. Input

2.1 Function call

3. Output

Program Output

21.4.3 `istream` Üye Fonksiyonları `peek`, `putback` ve `ignore`

- **`ignore`** üye fonkisyonu
  - Karakterlerin belirlenmiş sayısına atlar(default -bir)
  - Belirlenmiş bir ayırtaç bulunduğunda durur(default - **`EOF`**, dosya sonuna atlar)
- **`putback`** üye fonksiyonu
  - **`get`** ile elde edilen önceki karakteri tekrar akışa yerleştirir
- **`peek`**
  - Akıştan bir sonraki karakteri (yoketmeden) gönderir

21.4.4 Type-Safe I/O

- **<<** ve **>>** operatörleri
  - Farklı tip verileri kabul etmek için yüklenir
  - Umulmadık veri ile karşılaşıldığında , error bayrağı oluşur
  - Program kontrolde kalır

21.5 `read`, `gcount` ve `write` ile Formatsız I/O

- **`read`** ve **`write`** üye fonksiyonları
  - Formatsız I/O
  - Bellekteki bir karakter dizisine/dizisinden I/O ham byte-ı
  - Veri formatsız olduğundan, fonksiyonlar **`newline`** karakterinde durmaz. Örneğin
    - **`getline`** nın aksine belirlenmiş karakter sayısına kadar devam eder
  - Belirlenmiş karakter sayısından daha azı okunduğunda, failbit kurulur
- **`gcount`**:
  - Son girdi operasyonunda okunan toplam girdi karakteri sayısını gönderir

21.5 read, gcount ve write ile Formatsız I/O

```
1  // Fig. 21.15: fig21_15.cpp
2  // Formatsız I/O - read, gcount ve write ile
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  int main()
10 {
11    const int BOYUT = 80;
12    char destek[ BOYUT ];
13
14    cout << "Bir cümle gir:\n";
15    cin.read( destek, 20 );
16    cout << "\nGirilen cümle:\n";
17    cout.write( destek, cin.gcount() );
18    cout << endl;
19    return 0;
20 }
```

1. Initialize objects

2. Input

3. Output

Program Output

```
Bir cümle gir:
read, write, ve gcount üye fonksiyonlarının kullanımı
Girilen cümle:
read, write, ve gcou
```

- Stream düzenleyici kapasiteleri
  - Alan uzunluğu kurma
  - Duyarlılık kurma
  - Format bayraklarını kurma ve bozma
  - Alandaki doldurma karakterlerini kurma
  - Akışı boşaltma
  - Çıktı akışına yenisatır karakteri ekleme ve akışı boşaltma
  - Çıktı akışaına bir boş karakter ekleme ve girdi akışındaki whitespace karakterini atlama

21.6.1 Integral Stream Bazı: dec, oct, hex ve setbase

- **oct, hex,** veya **dec**:
  - Akıştaki tamsayıların bazını değiştirme.

  Örnek:

```
int n = 15;
 cout << hex << n;
```

  - "**F**" yazar

- **setbase**:
  - Tamsayı çıktısının bazını değiştirir
  - **<iomanip>** yükle
  - Bir tamsayı argüment kabul eder (**10**, **8**, veya **16**)

```
cout << setbase(16) << n;
```

  - Parametreleştirlmiş stream düzenleyici – bir argüment alır

```
1  // Fig. 21.16: fig21 16.cpp
2  // hex, oct, dec ve setbase stream düzenleyici kullanımı.
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  #include <iomanip>
10
11 using std::hex;
12 using std::dec;
13 using std::oct;
14 using std::setbase;
15
16 int main()
17 {
18    int n;
19
20    cout << "Bir tamsayı gir: ";
21    cin >> n;
22
23    cout << n << " hexadecimal değeri: "
24         << hex << n << '\n'
25         << dec << n << " oktal değeri: "
26         << oct << n << '\n'
27         << setbase( 10 ) << n << " desimal değeri: "
28         << n << endl;
29
30    return 0;
31 }
```

1. Load header

1.1 Initialize variables

2. Input number

3. Output in hex

3.1 Output in octal

3.2 Output in decimal

21.6.1 Integral Stream Bazı: dec, oct, hex ve setbase

```
Bir tamsayı gir: 20
20 hexadecimal değeri: 14
20 oktal değeri: 24
20 desimal değeri: 20
```

21.6.2 Floating-Point Duyarlılık (`precision, setprecision`)

- **`precision`**
  - Üye fonksiyonu
  - Desimal nokta sayısını kurar

    `cout.precision(2);`
  - **`cout.precision()`** o anki duyarlılık değerini gönderir
- **`setprecision`**
  - Parametreleştirlmiş stream düzenleyici
  - Tüm Parametreleştirlmiş stream düzenleyiciler gibi , **`<iomanip>`** gerekli
  - Duyarlılığı belirler:

    `cout << setprecision(2) << x;`
- Her iki yöntem için de, yeni değer kurulana kadar eskisi geçerli kalır

21.6.3 Alan Uzunluğu(Field Width) (`setw, width`)

- **`ios uzunluk`** üye fonksiyonu
  - Alan uzunluğunu kurar (çıktı karakterlerinin yer sayısı veya girdi karakterlerinin sayı)
  - Önceki uzunluğu gönderir
  - İşlenen değerler uzunluktan daha kısa ise, karakterler sağa yaslanarak doldurulur
  - Değerler kesilmez- tüm rakamlar basılır
  - **`cin.width(5);`**
- **`setw`** stream düzenleyici

```
cin >> setw(5) >> string;
```

- Boşluk karakteri için yer ayırmayı unutma!

```cpp
// fig21_18.cpp
// width(uzunluk) üye fonksiyonu kullanımı
#include <iostream>

using std::cout;
using std::cin;
using std::endl;

int main()
{
   int w = 4;
   char string[ 10 ];

   cout << "Bir cümle gir:\n";
   cin.width( 5 );

   while ( cin >> string ) {  // durmak için EOF gir
      cout.width( w++ );
      cout << string << endl;
      cin.width( 5 );
   }

   return 0;
}
```

, width)

1. Initialize variables

2. Input sentence

2.1 Set width

2.2 Loop and change width

3. Output

21.6.3 Alan Uzunluğu(Field Width) (`setw, width`)

```
Bir cümle gir:
Bu bir width üye fonksiyonu testidir
  Bu
  bir
  widt
      h
     üye
     fonk
      siyo
         nu
        test
         idir
^z
```

21.6.4 Kullanıcı Tanımlı Manipulatörler

- Kendi stream düzenleyicimizi oluşturabiliriz
  - **`bell`**
  - **`ret`** (taşıma dönüşü)
  - **`tab`**
  - **`endLine`**

- Parametreleştirilmiş stream düzenleyici
  - Kurulum klavuzuna bakınız

Format bayrakları (flag)

Stream I/O operasyonları sırasında uygulanacak formatı belirler

`setf`, `unsetf` ve `flags`

Bayrak kurumlarını kontrol eden üye fonksiyonlar

21.7.1 Format Durum Bayrakları

- Format Durum Bayrakları
  - **`ios`** sınıfında bir numaralandırma olarak tanımlanır
  - Üye fonksiyonlar tarafından kontrol edilebilir
  - **`flags`** – tüm bayrakların durumunu temsil eden bir değer belirler
    - İlk seçimleri içeren bir **`long`** değer gönderir
  - **`setf`** – tek parametre- bayrak kur
  - **`unsetf`** – bayrakları kaldırır
  - **`setiosflags`** - parametreleştirilmiş stream düzenleyici – bayrakları kurar
  - **`resetiosflags`** – parametreleştirilmiş stream düzenleyici, **`unsetf`** gibi aynı fonksiyonlara sahip
- Bayraklar bitwise OR (**`|`**) kullanılarak birleştirlebilir

21.7.2 Uç sıfırlar ve Desimal noktalar (`ios::showpoint`)

- **`ios::showpoint`**
  - Tamsayılı bir float sayının desimal değerlerini sıfırları ile birlikte yazar

    `cout.setf(ios::showpoint)`

    `cout << 79;`

    `79` sayısı **`79.00000`** olarak yazılır
    - Sıfırların sayısı duyarlılık kurulumu ile belli olur

21.7.3 Sağlama (`ios::left`, `ios::right`, `ios::internal`)

- **`ios::left`**
  - Sola yaslayarak yazar -fazlalık alan sağda boşluk olur
- **`ios::right`**
  - Default kurulum
  - Sağa yaslayarak yazar
- Karakter yaslamaları için
  - **`fill`** üye fonksiyonu
  - **`setfill`** parametrik akış düzenleyici
  - Default karakter -boşluk

- **internal** flag (bayrak)
  - Sayının işareti sola-yaslanmış(left-justified)
  - Sayının büyüklüğü sağa-yaslanmış(right-justified)
  - Karakter doldurma ile boşluklara müdahale

- **static** veri üyesi **ios::adjustfield**
  - **left**, **right** ve **internal** flag ları içerir
  - **left**, **right** veya **internal** sağlama bayrakları kurulduğunda, **ios::adjustfield** **setf** e ikinci argüment olmalı

```
cout.setf( ios::left, ios::adjustfield);
```

```
1  // Fig. 21.22: fig21_22.cpp
2  // sola yaslama sağa yaslama
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  #include <iomanip>
9
10 using std::ios;
11 using std::setw;
12 using std::setiosflags;
13 using std::resetiosflags;
14
15 int main()
16 {
17    int x = 12345;
18
19    cout << "Default sağa yaslama:\n"
20         << setw(10) << x << "\n\nÜve fonksiyonları kullanarak"
21         << "\n ios::left i kurmak için setf kullan:\n" << setw(10);
22
23    cout.setf( ios::left, ios::adjustfield );
24    cout << x << "\ndefault u geri yüklemek için unsetf kullan:\n";
25    cout.unsetf( ios::left );
26    cout << setw( 10 ) << x
27         << "\n\nparametrik akış düzenlevici kullanarak"
28         << "\nios::left i kurmak için setiosflags kullan:\n"
29         << setw( 10 ) << setiosflags( ios::left ) << x
30         << "\ndefault u geri yüklemek için resetiosflag kullan:\n"
31         << setw( 10 ) << resetiosflags( ios::left )
32         << x << endl;
33    return 0;
34 }
```

1. Initialize variable

2. Use parameterized stream manipulators

3. Output

21.7.3 Sağlama (ios::left, ios::right, ios::internal)

```
Default sağa yaslama:
     12345

Üye fonksiyonları kullanarak
ios::left i kurmak için setf kullan:
12345
default u geri yüklemek için unsetf kullan:
     12345

parametrik akış düzenleyici kullanarak
ios::left i kurmak için setiosflags kullan:
12345
default u geri yüklemek için resetiosflag kullan:
     12345
```

21.7.4 Tamponlama (`fill`, `setfill`)

- **`fill`** üye fonksiyonu
  - Doldurma karakterini belirler
  - default - boşluk
  - Önceki yaslama (tamponlama) karakterine döner

```
cout.fill( '*');
```

- **`setfill`** düzenleyici
  - Ayrıca doldurma karakterini kurar

```
cout << setfill ('*');
```

```
1  // Fig. 21.24: fig21_24.cpp
2  // fill ve setfill üye fonksiyonu kullanımı
3  // Yazılacak değerden daha geniş olan alanlar için
4  // doldurma karakterini değiştiren düzenleyici
5  #include <iostream>
6
7  using std::cout;
8  using std::endl;
9
10 #include <iomanip>
11
12 using std::ios;
13 using std::setw;
14 using std::hex;
15 using std::dec;
16 using std::setfill;
17
18 int main()
19 {
20    int x = 10000;
```

ill)

1. Load header

1.1 Initialize variable

```
21
22     cout << x << " in sağ ve sol yaslamalı yazılışı\n"
23          << "ayrıca iç düzenlemeli hex yazılışı.\n"
24          << "default doldurma karakteri(boşluk)kullanarak:\n";
25     cout.setf( ios::showbase );
26     cout << setw( 10 ) << x << '\n';
27     cout.setf( ios::left, ios::adjustfield );
28     cout << setw( 10 ) << x << '\n';
29     cout.setf( ios::internal, ios::adjustfield );
30     cout << setw( 10 ) << hex << x;
31
32     cout << "\n\nDeğişik doldurma karakterleri ile:\n";
33     cout.setf( ios::right, ios::adjustfield );
34     cout.fill( '*' );
35     cout << setw( 10 ) << dec << x << '\n';
36     cout.setf( ios::left, ios::adjustfield );
37     cout << setw( 10 ) << setfill( '%' ) << x << '\n';
38     cout.setf( ios::internal, ios::adjustfield );
39     cout << setw( 10 ) << setfill( '^' ) << hex << x << endl;
40     return 0;
41 }
```

```
10000 in sağ ve sol yaslamalı yazılışı
Ayrıca iç düzenlemeli hex yazılışı.
Default doldurma karakteri(boşluk) kullanarak:
     10000
10000
0x    2710

Değişik doldurma karakterleri ile:
*****10000
10000%%%%%
0x^^^^2710
```

ill)

2. Set fill character

3. Output

Program Output

- **ios::basefield** static üye
  - **setf** ile **ios::adjustfield** e benzer şekilde kullanım
  - **ios::oct**, **ios::hex** ve **ios::dec** flag bit-lerini içerir
  - Tamsayıların oktal, hexadecimal ve desimal değerler olarak ele alınmasını belirler
  - Default –desimal
  - stream aktarmalar için default girilen forma bağlıdır
    - 0 ile başlayan tamsayılar oktal kabul edilir
    - **0x** veya **0X** ile başlayan hexadecimal
  - Bir baz belirlendikten sonra , kurulum değiştirilinceye kadar aynı kalır

- **`ios::scientific`**
  - Reel sayının çıktısını bilimsel notasyonda verir:
    - `1.946000e+009`

- **`ios::fixed`**
  - Desimalin sağında belli sayıda rakam göstermeye zorlar (**`precision`** ile belirlenir)

- **`static`** veri üyesi **`ios::floatfield`**
  - **`ios::scientific`** ve **`ios::fixed`** içerir
  - **`setf`** deki **`ios::adjustfield`** ve **`ios::basefield`** de benzer şekilde kullanım
    - **`cout.setf(ios::scientific, ios::floatfield);`**
  - **`cout.setf(0, ios::floatfield)`** reel sayı çıktıları için default formata geçer

```cpp
// Fig. 21.26: fig21_26.cpp
// sistem default, bilimsel ve sabitlenmiş formatta
// reel sayılar.
#include <iostream>

using std::cout;
using std::endl;
using std::ios;

int main()
{
   double x = .001234567, y = 1.946e9;

   cout << "default format ile göster:\n"
        << x << '\t' << y << '\n';
   cout.setf( ios::scientific, ios::floatfield );
   cout << "bilimsel format ile göster:\n"
        << x << '\t' << y << '\n';
   cout.unsetf( ios::scientific );
   cout << "unsetf den sonra default formatta göster:\n"
        << x << '\t' << y << '\n';
   cout.setf( ios::fixed, ios::floatfield );
   cout << "sabitlenmiş format ile göster:\n"
        << x << '\t' << y << endl;
   return 0;
}
```

```
default format ile göster:
0.00123457      1.946e+009
bilimsel format ile göster:
1.234567e-003   1.946000e+009
unsetf den sonra default formatta göster:
0.00123457      1.946e+009
sabitlenmiş format ile göster:
0.001235        1946000000.000000
```

ific, ios::fixed)

1. Initialize variables

2. Set flags

3. Output

Program Output

21.7.7 Uppercase/Lowercase Kontrol (`ios::uppercase`)

- **`ios::uppercase`**
  - Bilimsel notasyonda **`E`** gösterimi

    `4.32E+010`
  - hexadecimal sayıların **`X`** ile gösterimi –tüm harf değerler büyük harf olur

    `75BDE`

- **`flags`** üye fonksiyonu
  - Format bayrağının o anki değerini argümentsiz gönderir (**`long`** değer olarak)
  - **`long`** argumenti ile, format bayrağını belirlendiği gibi kurar
    - Ön kurulumları gönderir
- **`setf`** üye fonksiyonu
  - Argümentinde verilen format bayraklarını kurar
  - Önceki flag kurulumunu bir **`long`** değer olarak gönderir

```
long previousFlagSettings =
   cout.setf( ios::showpoint | ios::showpos );
```

- İki **`long`** argument ile **`setf`**

```
cout.setf( ios::left, ios::adjustfield );
```

`ios::adjustfield` bitlerini temizler **`ios::left`** kurar

  - setf nin bu versiyonu aşağıdakiler ile kullanılabilir:
  - **`ios::basefield (ios::dec, ios::oct, ios::hex)`**
  - **`ios::floatfield (ios::scientific, ios::fixed)`**
  - **`ios::adjustfield (ios::left, ios::right, ios::internal )`**

- **`unsetf`**
  - Belirlenmiş bayrakları yeniden kurar
  - Önceki kurumlara döner

```
1  // Fig. 21.28: fig21 28.cpp
2  // flag üye fonksiyonları kullanımı.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7  using std::ios;
8
9
10 int main()
11 {
12    int i = 1000;
13    double d = 0.0947628;
14
15    cout << "flag değişkeninin değeri: "
16         << cout.flags()
17         << "\norjinal formatta  int ve double yaz:\n"
18         << i << '\t' << d << "\n\n";
19    long originalFormat =
20            cout.flags( ios::oct | ios::scientific );
21    cout << "flag değişkeninin değeri: "
22         << cout.flags()
23         << "\nyeni formatta int ve double yazımı-\n"
24         << "flag üye fonksiyonu kullanılarak belirlenmiş:\n"
25         << i << '\t' << d << "\n\n";
26    cout.flags( originalFormat );
27    cout << "flag değişkeninin değeri: "
28         << cout.flags()
29         << "\ndeğerleri tekrar orijinal formatta yaz:\n"
30         << i << '\t' << d << endl;
31    return 0;
32 }
```

1. Initialize variables

2. Set flags

3. Output

21.7.8 Format bayraklarını kurma ve yenileme(flags, setiosflags, resetiosflags)

```
flag değişkeninin değeri: 0
original formatta int ve double yaz:
1000    0.0947628

flag değişkeninin değeri: 4040
yeni formatta int ve double yazımı
flag üye fonksiyonu kullanılarak belirlenmiş:
1750    9.476280e-002

flag değişkeninin değeri : 0
değerleri tekrar orijinal formatta yaz:
1000    0.0947628
```

21.8 Stream Hata Durumları

- **`eofbit`**
  - EOF görüldükten sonra bir girdi akışı kurar
  - Eğer **`cin`** de EOF ile karşılaşılırsa **`cin.eof() true`** gönderir

- **`failbit`**
  - Format hatası oluştuğunda bir akış kurulur
  - Eğer akış operasyonu başarısız olursa **`cin.fail() true`** gönderir
  - Normal durumlarda bu hatalardan kurtulunur

21.8 Stream Hata Durumları

- **`badbit`**
  - Veri kaybı ile sonuçlanan hata oluştuğunda kurulur
  - Akış operasyonu başarısız ise, **`cin.bad()`** **`true`** gönderir
  - Geri dönüşü yok
- **`goodbit`**
  - **`eofbit`**, **`failbit`** veya **`badbit`** kurulmadığında, bir akış için kurulur
  - Eğer **`bad`**, **`fail`** ve **`eof`** tümü false gönderecekse, **`cin.good()`** **`true`** gönderir.
  - I/O operasyonları sadece "good" akışlarda kullanılmalı
- **`rdstate`**
  - Akışın durumunu gönderir
  - Akış tüm durum bitlerini inceleyen bir **`switch`** deyimiyle test edilmelidir
  - Durumu belirlemek için **`eof`**, **`bad`**, **`fail`**, ve **`good`** kullanmak daha kolaydır

21.8 Stream Hata Durumları

- **`clear`**
  - Akışın durumunu "good" a çevirmek için kullanılır
  - **`cin.clear()`** **`cin`** i temizler ve akış için **`goodbit`** kurar
  - **`cin.clear( ios::failbit )`** aslında **`failbit`** kurar
    - Kullanıcı tanımlı tiplerde hata oluştuğunda bu kullanılabilir
- Diğer operasyonlar
  - **`operator!`**
    - Eğer **`badbit`** veya **`failbit`** kuruluysa **`true`** gönderir
  - **`operator void*`**
    - Eğer **`badbit`** veya **`failbit`** kuruluysa **`false`** gönderir
  - Dosya işleme için kullanışlı

```
1  // Fig. 21.29: fig21_29.cpp
2  // Hata durumlarını test etme.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7  using std::cin;
8
9  int main()
10 {
11    int x;
12    cout << "Kötü bir girdi operasyonundan önce:"
13         << "\ncin.rdstate(): " << cin.rdstate()
14         << "\n    cin.eof(): " << cin.eof()
15         << "\n   cin.fail(): " << cin.fail()
16         << "\n    cin.bad(): " << cin.bad()
17         << "\n   cin.good(): " << cin.good()
18         << "\nBir tamsayı beklivor fakat karakter girilivor: ";
19    cin >> x;
20
21    cout << "\nKötü bir girdi operasyonundan sonra:"
22         << "\ncin.rdstate(): " << cin.rdstate()
23         << "\n    cin.eof(): " << cin.eof()
24         << "\n   cin.fail(): " << cin.fail()
25         << "\n    cin.bad(): " << cin.bad()
26         << "\n   cin.good(): " << cin.good() << "\n\n";
27
28    cin.clear();
29
30    cout << " cin.clear()den sonra"
31         << "\ncin.fail(): " << cin.fail()
32         << "\ncin.good(): " << cin.good() << endl;
33    return 0;
34 }
```

1. Initialize variable

2. Function calls

3. Output

21.8 Stream Hata Durumları

```
Kötü bir girdi operasyonundan önce:
cin.rdstate(): 0
    cin.eof(): 0
   cin.fail(): 0
    cin.bad(): 0
   cin.good(): 1

Bir tamsayı bekliyor fakat karakter giriliyor: A

Kötü bir girdi operasyonundan sonra:
cin.rdstate(): 2
    cin.eof(): 0
   cin.fail(): 1
    cin.bad(): 0
   cin.good(): 0

cin.clear() den sonra
cin.fail(): 0
cin.good(): 1
```

21.9 Output Stream-i Input Stream-e bağlama

- **`tie`** üye fonksiyonu
  - bir **`istream`** ve bir **`ostream`** operasyonunu senkronize eder
  - Çıktı ardışık girdiden önce görünür
  - **`cin`** ve **`cout`** için otomatik olarak yapılır
- **`inputStream.tie( &outputStream );`**
  - **`inputStream`** e **`outputStream`** e bağlar
  - **`cin.tie( &cout)`** otomatik yapılır
- **`inputStream.tie( 0 );`**
  - Bir çıktı akışında **`inputStream`** i çözer (bağını koparır)