C++

Ders 8

Stream ve Şablon Yapıları

Prof. Dr. M. Serdar ÇELEBİ, İTÜ

C++

Stream'ler

- Stream
  - Byte'ların dizisi şeklinde verinin transferidir
- G/Ç 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 stream.

C++

Stream'ler

- G/Ç operasyonları engel teşkil edebilir:
  - Bir stream'in akması için gereken süre CPU'nun stream'deki veriye işlem yapması için gereken zamandan kat kat fazladır.
- Düşük düzeyde G/Ç
  - formatsız
  - byte birim olarak ilgilenme
  - yüksek hız, fakat elverişsizdir

C++

Stream'ler

- İleri Düzeyde G/Ç
  - Formatlı
  - Anlamlı birimlere gruplandırılmış byte'lar: tamsayılar, karakterler, vs.
  - Yüksek miktarda dosya işlemleri dışında bütün G/Ç için iyidir

C++

`iostream` Kütüphanesi Başlık Dosyaları

- **`iostream`** kütüphanesi:
  - **`<iostream.h>`: `cin`, `cout`, `cerr`,** ve **`clog`** nesnelerini içerir
  - **`<iomanip.h>`:** parametreleştirilmiş stream manipulatörlerini kapsar
  - **`<fstream.h>`:** Kullanıcı kontrollü dosya işlemleri için önemli olan bilgileri içerir.

C++

Stream Giriş/Çıkış Sınıfları ve Nesneleri

- **`ios`:**
  - **`istream`** ve **`ostream`** , **`ios'dan miras almıştır`**
  - **`iostream`** , **`istream`** ve **`ostream'den miras almıştır`**.
- **<<** (sola kaydırma operatörü): *stream araya sokma (insertion) operatörü* olarak aşırı yüklenmiştir
- **>>** (sağa kaydırma operatörü): *stream araya sokma (extraction) operatörü* olarak aşırı yüklenmiştir.
- **`cin`**, **`cout`**, **`cerr`**, **`clog`** ve kullanıcı tanımlı stream nesneleriyle birlikte kullanılır.

C++

Stream Giriş/Çıkış Sınıfları ve Nesneleri

- **`istream:`** stream girişini sağlar

  `cin >> someVariable`;

  - **`cin, someVariable`**'a hangi tür bir değişken atandığını biliyor (**`someVariable`**'ın türüne dayandırılarak).
- **`ostream:`** stream çıkışını sağlar.
  - **`cout << someVariable;`**
    - cout, hangi türden bir verinin çıktı olacağını biliyor
  - **`cerr << someString;`**
    - Tamponlanmamış (Unbuffered). **`someString`** 'i hemen yazar.

C++

Stream Giriş/Çıkış Sınıfları ve Nesneleri

- **`ostream:`** stream cıkışını sağlar
  - **`clog << someString;`**
    - Tamponlanmış (Buffered). **`someString`**'i buffer dolar dolmaz ya da boşaltılmak istendiğinde yazar.
- **`ostream`**: formatlı ve formatsız çıktı işlemlerini gerçekleştirir
  - karakterler için **`put`** ve formatsız karakterler için **`write`** kullanılır
  - Sayıların 10'luk, 8'lik ve 16'lık düzende yazılışı
  - Kayan noktalı sayıların virgülden sonraki hassasiyetleri
  - Formatlı metin çıktıları.

C++

Stream-Araya Sokma (Insertion) Operatorü

- **<<** : varolan türlerin çıktıları için aşırı yüklenmiştir:
  - Kullanıcı tanımlı türlerin çıktıları için de kullanılabilir.
  - **`cout << '\n';`**
    - newline character basar (alt satıra götürür imleci)
  - **`cout << endl;`**
    - **`endl`** bir stream manipulatörüdür ve newline karakteriyle aynı işi yapar ve çıkış buffer boşaltılır
  - **`cout << flush;`**
    - **`flush`** çıkış buffer boşaltılır

C++

Tekrarlı Stream Insertion / Extraction

- **<<** : soldan sağa işlem uygulanır ve solundaki işlemci nesneye bir referans döndürür (yani **cout**'a).
- Aşağıdaki basamaklı uygulama

```
cout << "Merhaba" << " bugün" << "
  nasılsın?";
```

- Parantez kullanmayı unutma:

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

- Alttaki gibi değil!

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

C++

`char *` Değişkenlerinin Çıkışı

- `<< char *` türünden bir değişkeni string gibi basar
- Bir string'in ilk karakterinin adresini basmak için `void *` kullanarak tür değişikliği yapmak lazım.

C++

char * Değişkenlerinin Çıkışı

```
1  // Fig. 11.8: fig11 08.cpp
2  // Printing the address stored in a char* variable
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  int main()
9  {
10    char *string = "test";
11
12    cout << "Value of string is: " << string
13         << "\nValue of static cast< void * >( string ) is: "
14         << static cast< void * >( string ) << endl;
15    return 0;
16 }
```

```
Value of string is: test
Value of static_cast< void *>( string ) is: 0046C070
```

C++

Üye Fonksiyon `put` ile Karakter Çıkışı

- **`put`** üye fonksiyonu
  - belirtilen streame bir karakter çıktısı verir

    ```
    cout.put( 'A');
    ```
  - Kendisini çağıran nesneye referans dönderdiği için basamaklı şekilde çağrılabilir

    ```
    cout.put( 'A' ).put( '\n' );
    ```
  - ASCII değerli bir ifade ile çağrılabilir

    ```
    cout.put(65);
    ```

    A çıktısını verir.

C++

Stream Girişi

- **>>** (stream-çıkış (extraction))
  - stream giriş işlemleri için kullanılır
  - Beyaz boşlukları (boşluklar, tab, yeni-satır) normalde ihmal eder
  - EOF ile karşılaştığında 0 (yanlış) döndürür, aksi takdirde çağrılan nesneye referans döndürür (mesela **`cin`**)
  - Bu basamaklı giriş işlemlerine imkan verir.

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

- **>>** stream'in bit durumunu kontrol eder
  - yanlış türden veri girişi için **`failbit`**
  - operasyon gerçekleşmezse **`badbit`**

C++

Stream-Çıkış (Extraction) Operatorü

- `>>` ve `<<` diğerlerine göre yüksek önceliğe sahiptir
  - şartlı ifadeler ve aritmetik işlemler parantez içinde yazılmalıdır

- döngüleri yapmak için popüler bir yoldur

```
while (cin >> grade)
```

  - eğer EOF ile karşılaşılırsa çıkarma (extraction) 0 (yanlış) döndürür ve döngü biter

C++

Stream-Çıkış (Extraction) Operatorü

```
1  // Fig. 11.11: fig11_11.cpp
2  // Stream-extraction operator returning false on end-of-file.
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  int main()
10 {
11    int grade, highestGrade = -1;
12
13    cout << "Enter grade (enter end-of-file to end): ";
14    while ( cin >> grade ) {
15       if ( grade > highestGrade )
16          highestGrade = grade;
17
18       cout << "Enter grade (enter end-of-file to end): ";
19    }
```

C++

Stream-Çıkış (Extraction) Operatorü

```
20
21     cout << "\n\nHighest grade is: " << highestGrade << endl;
22     return 0;
23 }
```

```
Enter grade (enter end-of-file to end): 67
Enter grade (enter end-of-file to end): 87
Enter grade (enter end-of-file to end): 73
Enter grade (enter end-of-file to end): 95
Enter grade (enter end-of-file to end): 34
Enter grade (enter end-of-file to end): 99
Enter grade (enter end-of-file to end): ^Z
Highest grade is: 99
```

C++

`get` ve `getline` Üye Fonksiyonları

- **`cin.get()`:**
  - streamden bir karakter girişi yapar (beyaz boşluklar da dahil) ve onu döndürür
- **`cin.get( c )`:**
  - streamden bir karakter girişi yapar ve onu c'de saklar.
- **`cin.get(array, size)`:**
  - 3 değişken alır: karakterlerin dizisi, boy limiti, ve bir sınırlayıcı (delimiter) (varsayılan **`\n`**)
  - diziyi tampon (buffer) gibi kullanır

C++

`get` ve `getline` Üye Fonksiyonları

- **`cin.get(array, size)`:**
  - sınırlayıcıyla karşılaşıldığında giriş streaminde kalır
  - Null karakter diziye sokulur
  - dizi streamden boşaltılmadıkça orda kalır
- **`cin.getline(array, size)`**
  - **`cin.get(buffer, size)`** gibi çalışır, fakat sınırlayıcıyı streamden atar ve diziye koymaz
  - Null karakter diziye sokulur

C++

get ve getline Üye Fonksiyonları

```
1  // Fig. 11.12: fig11_12.cpp
2  // Using member functions get, put and eof.
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 << "Before input, cin.eof() is " << cin.eof()
14         << "\nEnter a sentence followed by end-of-file:\n";
15
16    while ( ( c = cin.get() ) != EOF )
17       cout.put( c );
```

cin.eof() **yanlış** ise **(0)** ya da **doğruy** ise **(1)** döndürür

cin.get() girdi stream'indeki bir sonraki karakteri döndürür, boşluklar da dahil.

C++

get ve getline Üye Fonksiyonları

```
18
19     cout << "\nEOF in this system is: " << c;
20     cout << "\nAfter input, cin.eof() is " << cin.eof() << endl;
21     return 0;
22 }
```

```
Before input, cin.eof() is 0
Enter a sentence followed by end-of-file:
Testing the get and put member functions^Z
Testing the get and put member functions
EOF in this system is: -1
After input cin.eof() is 1
```

C++

get ve getline Üye Fonksiyonları

```
1  // Fig. 11.14: fig11_14.cpp
2  // Character input with member function getline.
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  int main()
10 {
11    const SIZE = 80;
12    char buffer[ SIZE ];
13
14    cout << "Enter a sentence:\n";
15    cin.getline( buffer, SIZE );
16
17    cout << "\nThe sentence entered is:\n" << buffer << endl;
18    return 0;
19 }
```

```
Enter a sentence:
Using the getline member function

The sentence entered is:
Using the getline member function
```

C++

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

- **`ignore`** üye fonksiyonu
  - verilen sayı kadar karakteri atlar (varsayılan bir)
  - verilen sınırlandırıcıyla karşılaşıldığında işlemi durdurur (varsayılan **`EOF`**, dosyanın sonu)
- **`putback`** üye fonksiyonu
  - **`get`** ile alınan bir önceki karakteri streame geri koyar
- **`peek`**
  - streamdeki bir sonraki karakteri onu silmeden dönderir

C++

Tip Korumalı G/Ç

- **<<** ve **>>** operatörleri
  - değişik türde verileri alabilmek için aşırı yüklenmiştir
  - beklenilmedik bir veriyle karşılaşıldığında hata bayrakları ayarlanır
  - program kontrol altında olur.

C++

`read`, `gcount` ve `write` ile formatsız G/Ç

- **`read`** ve **`write`** üye fonksiyonları
  - Formatsız G/Ç
  - Hafızada olan bir karakter dizinine veya dizininden işleme sokulmamış byte'ların G/Ç işlemleri
  - Veriler formatsız olduğu için, mesela **`newline`** 'da fonksiyon durmayacaktır.
  - Yerine, **`getline`**'daki gibi, belirtilen karakter sayısı kadar işlem yapacaklar.
  - Eğer verilen değerden az karakter okunduysa failbit ayarlanır
- **`gcount`**:
  - en son giriş işleminde okunan toplam karakter sayısını döndürür.

C++

read, gcount ve write ile Formatsız G/Ç

```
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  int main()
10 {
11    const int SIZE = 80;
12    char buffer[ SIZE ];
13
14    cout << "Enter a sentence:\n";
15    cin.read( buffer, 20 );
16    cout << "\nThe sentence entered was:\n";
17    cout.write( buffer, cin.gcount() );
18    cout << endl;
19    return 0;
20 }
```

Sadece ilk **20** karakteri okur

g.count() 20'yi dönderir, çünkü en son input işlemiyle okunan karakter sayısı 20.

```
Enter a sentence:
Using the read, write, and gcount member functions
The sentence entered was:
Using the read, writ
```

C++

Stream Manipulatörleri

- stream manipulatör yetenekleri:
  - Alan ayarları
  - Hassasiyet ayarları
  - Format bayraklarının ayarları veya sıfırlaması
  - Alanları doldurma karakteri ayarı
  - Stream'i boşaltma
  - Çıkış stream'ine yeni bir satır ekleme ve stream'i boşaltma çıkış stream'ine null karakter ekleme ve giriş stream'indeki beyaz boşlukları ihmal etme

C++

Integral Stream Tabanı: `dec`, `oct`, `hex` ve `setbase`

- **`oct`, `hex`,** veya **`dec`**:
  - Stream'de belirtilene göre tabanı değiştirme.

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

  - `"F"` basılır
- **`setbase`**:
  - çıkış tamsayısının tabanını değiştirir
  - `<iomanip>` başlık dosyası yüklenmeli
  - değişken olarak tamsayı kabul eder (`10`, `8`, `16`)

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

  - parametreleştirilmiş stream manipulatörü bir değişken alır

C++

Integral Stream Tabanı: dec,oct,hex ve setbase

```
1  // Fig. 11.16: fig11_16.cpp
2  // Using hex, oct, dec and setbase stream manipulators.
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
```

C++

Integral Stream Tabanı: dec,oct,hex ve setbase

```
20     cout << "Enter a decimal number: ";
21     cin >> n;
22
23     cout << n << " in hexadecimal is: "
24          << hex << n << '\n'
25          << dec << n << " in octal is: "
26          << oct << n << '\n'
27          << setbase( 10 ) << n << " in decimal is: "
28          << n << endl;
29
30     return 0;
31 }
```

Enter a decimal number: 20

20 in hexadecimal is: 14

20 in octal is: 24

20 in decimal is: 20

```
Enter a decimal number: 20
20 in hexadecimal is: 14
20 in octal is: 24
20 in decimal is: 20
```

C++

Kayan Nokta Hassasiyeti (`precision, setprecision`)

- **`precision`**
  - Üye fonksiyon
  - Ondalıklı kısmın sağında kaç sayı yazılacağını belirler **`cout.precision(2);`**
  - **`cout.precision()`** geçerli hassasiyet ayarını dönderir
- **`setprecision`**
  - parametreleştirilmiş stream manipulatörü
  - Bütün parametreleştirilmiş stream manipulatörleri gibi, **`<iomanip>`** yüklenmelidir
  - Hassasiyet ayarı yapılır:

    `cout << setprecision(2) << x;`
- Her iki metod için yeni bir değer verilinceye kadar eski ayar geçerlidir.

C++

Alan Genişliği (`setw,width`)

- **ios width** üye fonksiyonları
  - Alan genişliğini ayarlar (giriş-çıkış işlemlerinde karakterlerin yazılacağı).
  - Bir önceki genişliği dönderir.
  - Eğer verilen değer genişlikten küçükse doldurma karakteri boşlukları doldurmak için kullanılır ama değerler kesilmez, sayı büyükse bütün sayı yazılır
  - **cin.width(5);**
- **setw** stream manipulatörü

  cin >> setw(5) >> string;
- null karakter için bir yer ayırmayı unutma

C++

Alan Genişliği (setw,width)

```
1  // fig11 18.cpp
2  // Demonstrating the width member function
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  int main()
10 {
11    int w = 4;
12    char string[ 10 ];
13
14    cout << "Enter a sentence:\n";
15    cin.width( 5 );
16
17    while ( cin >> string ) {
18       cout.width( w++ );
19       cout << string << endl;
20       cin.width( 5 );
21    }
22
23    return 0;
24 }
```

C++

Alan Genişliği (`setw,width`)

```
Enter a sentence:
This is a test of the width member function
This
   is
     a
   test
      of
      the
      widt
          h
        memb
           er
          func
           tion
```

C++

Kullanıcı tanımlı Manipulatörler

- Kullanıcı kendi manipulatörlerini oluşturabilir
  - **bell**
  - **ret** (carriage return)
  - **tab**
  - **endLine**

- parametreleştirilmiş stream manipulatörleri kurulum rehberine danışır.

C++

Stream Format Durumları

- Format bayrakları
  - G/Ç işlemleri sırasında uygulanacak olan formatlama verilmeli

- **setf**, **unsetf** ve **flags**
  - bayrak ayarlarını kontrol eden üye fonksiyonlar

C++

Format Durumu Bayrakları

- Format Durum bayrakları
  - **`ios`** sınıfında sıralama olarak tanımlanmıştır
  - üye fonksiyonlar tarafından kontrol edilebilir
  - **`flags`** – bütün bayrakların ayarlarını temsil eden bir değer belirtir
    - Bir önceki seçenekleri kapsayan **`long`** değerini döndürür
  - **`setf`** – bir değişken, var olan bayrakları "or (|)" ile birleştirebilir.

C++

Format Durumu Bayrakları

- Format Durum bayrakları
  - **setf** – bir değişken, var olan bayrakları “or (|)” ile birleştirebilir
  - **unsetf –** bayrakları sıfırlar (yapılan ayarları geri alır)
  - **setiosflags** – bayrak ayarlarını yapmak için parametreleştirilmiş stream manipulatörü
  - **resetiosflags** – parametreleştirilmiş stream manipulatörü, **unsetf** ile aynı fonksiyona sahiptir
- Bayraklar bitwise or "|“ kullanılarak birleştirelebilir.

C++

Sıfırları ve Ondalıklı basamakları Yazdırma

- **`ios::showpoint`**
  - Bir ondaklı sayıyının ondalıklı kısmını sıfırlarıyla beraber yazılmasını sağlar

    `cout.setf(ios::showpoint)`

    `cout << 79;`

    `79 'u 79.00000 olarak basar`
  - Sıfırların sayısı hassasiyet ayarlarıyla belirlenir.

C++

Dayalı yazma (ios::left, ios::right, ios::internal)

- **`ios::left`**
  - sağa doldurmayla beraber sola dayalı yazmayı sağlar
- **`ios::right`**
  - Varsayılan ayar
  - sola doldurmayla beraber sağa dayalı yazmayı sağlar
- Doldurma karakteri ayarlamak için:
  - **`fill`** üye fonksiyonu
  - **`setfill`** parametreleştirilmiş stream manipulatörü kullanılır.
  - Varsayılan karakter boşluk karakteridir.

C++

Dayalı yazma (ios::left, ios::right, ios::internal)

- **`ios::internal`**
  - sayıların işareti sola dayalı
  - sayılar sağa dayalı
  - araya gelen boşluklar doldurma karakteri ile doldurulur
- **`static`** veri üye **`ios::adjustfield`**
  - **`left`**, **`right`** ve **`internal`** bayraklarını kapsar
  - **`ios::adjustfield`**, **`left`**, **`right`** veya **`internal`** bayrakları kullanılırsa , **`setf`**'ye ikinci değişken olmalı.

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

C++

Dayalı yazma

```
1  // Fig. 11.22: fig11_22.cpp
2  // Left-justification and right-justification.
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 is right justified:\n"
20         << setw(10) << x << "\n\nUSING MEMBER FUNCTIONS"
21         << "\nUse setf to set ios::left:\n" << setw(10);
```

```
Default is right justified:
     12345

USING MEMBER FUNCTIONS
Use setf to set ios::left:
12345
```

C++

Dayalı yazma

```
23    cout.setf( ios::left, ios::adjustfield );
24    cout << x << "\nUse unsetf to restore default:\n";
25    cout.unsetf( ios::left );
26    cout << setw( 10 ) << x
27         << "\n\nUSING PARAMETERIZED STREAM MANIPULATORS"
28         << "\nUse setiosflags to set ios::left:\n"
29         << setw( 10 ) << setiosflags( ios::left ) << x
30         << "\nUse resetiosflags to restore default:\n"
31         << setw( 10 ) << resetiosflags( ios::left )
32         << x << endl;
33    return 0;
34 }
```

```
Default is right justified:
     12345

USING MEMBER FUNCTIONS
Use setf to set ios::left:
12345
Use unsetf to restore default:
     12345

USING PARAMETERIZED STREAM MANIPULATORS
Use setiosflags to set ios::left:
12345
Use resetiosflags to restore default:
     12345
```

```
Use unsetf to restore default:
     12345
```

```
USING PARAMETERIZED STREAM MANIPULATORS
Use setiosflags to set ios::left:
12345
```

```
Use resetiosflags to restore default:
     12345
```

C++

Doldurma (fill, setfill)

- **`fill`** üye fonksiyonu
  - doldurma karakterini belirtir
  - Varsayılan boşluk karakteridir
  - bir önceki doldurma karakterini dönderir

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

- **`setfill`** manipulatörü
  - doldurma karakterini ayarlar

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

C++

Doldurma (fill, setfill)

```
1  // Fig. 11.24: fig11_24.cpp
2  // Using the fill member function and the setfill
3  // manipulator to change the padding character for
4  // fields larger than the values being printed.
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;
```

C++

Doldurma (fill, setfill)

```
21
22    cout << x << " printed as int right and left justified\n"
23         << "and as hex with internal justification.\n"
24         << "Using the default pad character (space):\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\nUsing various padding characters:\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 }
```

C++

Doldurma (fill, setfill)

```
10000 printed as int right and left justified
and as hex with internal justification.
Using the default pad character (space):
     10000
10000
0x    2710

Using various padding characters:
*****10000
10000%%%%%
0x^^^^2710
```

C++

Integral Stream Tabanı

- **`ios::basefield`** statik üye
  - **`setf`**'le **`ios::adjustfield`** kullanımına benzer.
  - **`ios::oct`**, **`ios::hex`** ve **`ios::dec`** bayrak bitlerini barındırır.
  - Tamsayıların 8'lik, 16'lık ve 10'luk değerlerinin kullanılacağını bildirir. Varsayılan 10'luk değerdir.
  - Varsayılan stream çıkarmaları (extraction) girilen forma bağlıdır
    - **`0`**'la başlayan tamsayılar 8'likmiş gibi davranılır
    - **`0x`** veya **`0X`**'le başlayan tamsayılar 16'lıkmış gibi davranılır
  - Bir taban ayarlandıktan sonra değiştirilene kadar

C++

Kayan Nokta Sayıları; Bilimsel Gösterilim

- **`ios::scientific`**
  - ondalıklı sayı bilimsel gösterilim biçiminde yazdırır.

    `1.946000e+009`

- **`ios::fixed`**
  - ondalıklı sayının ondalık kısmının sağında tanımlanan sayı kadar gösterilmesini sağlar (**`precision`** ile).

C++

Kayan Nokta Sayıları; Bilimsel Gösterilim

- **`static`** data üyesi **`ios::floatfield`**
  - **`ios::scientific`** ve **`ios::fixed`** 'i kapsar
  - **`setf`** içinde **`ios::adjustfield`** ve **`ios::basefield`** gibi kullanılır
    - **`cout.setf(ios::scientific, ios::floatfield);`**
  - **`cout.setf(0, ios::floatfield)`** kayan noktalı sayıların varsayılan çıkış özelliklerine geri dönmeyi sağlar.

C++

Kayan Nokta Sayıları; Bilimsel Gösterilim

```
1  // Fig. 11.26: fig11 26.cpp
2  // Displaying floating-point values in system default,
3  // scientific, and fixed formats.
4  #include <iostream>
5
6  using std::cout;
7  using std::endl;
8  using std::ios;
9
10 int main()
11 {
12    double x = .001234567, y = 1.946e9;
13
14    cout << "Displayed in default format:\n"
15         << x << '\t' << y << '\n';
16    cout.setf( ios::scientific, ios::floatfield );
17    cout << "Displayed in scientific format:\n"
18         << x << '\t' << y << '\n';
19    cout.unsetf( ios::scientific );
20    cout << "Displayed in default format after unsetf:\n"
21         << x << '\t' << y << '\n';
```

C++

Kayan Nokta Sayıları; Bilimsel Gösterilim

```
22     cout.setf( ios::fixed, ios::floatfield );
23     cout << "Displayed in fixed format:\n"
24          << x << '\t' << y << endl;
25     return 0;
26 }
```

```
Displayed in default format:
0.00123457      1.946e+009
Displayed in scientific format:
1.234567e-003   1.946000e+009
Displayed in default format after unsetf:
0.00123457      1.946e+009
Displayed in fixed format:
0.001235        1946000000.000000
```

C++

Büyük/Küçük Harf Kontrolü (ios::uppercase)

- **`ios::uppercase`**
  - Bilimsel gösterilimde büyük E yazılmasını sağlar

    `4.32E+010`
  - 16'lık düzende, sayılarda geçen harflerin ve tüm harflerin büyük yazılmasını sağlar

    `75BDE`

C++

Format Bayraklarını Ayarlama

- **`flags`** üye fonksiyonu
  - değişkensiz, geçerli olan format bayrak ayarlarını dönderir (**`long`** değer olarak)
  - **`long`** değişkenle, belirtilen format ayarını yapar
    - Bir önceki ayarları dönderir
- **`setf`** üye fonksiyonu
  - değişken olarak aldığı format bayrak ayarlarını yapar
  - **`long`** değer olarak önceki bayrak ayarlarını dönderir.

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

C++

Format Bayraklarını Ayarlama

- iki **`long`** değişkenli **`setf`**

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

  `ios::adjustfield` için gerekli bitleri temizler ve **`ios::left`** ayarını yapar.

  Bu setf versiyonu

  - **`ios::basefield (ios::dec, ios::oct, ios::hex)`**
  - **`ios::floatfield (ios::scientific, ios::fixed)`**
  - **`ios::adjustfield (ios::left, ios::right, ios::internal )`** ile beraber kullanılabilir.

- **`unsetf`**
  - belirtilen bayrakları sıfırlar (varsayılana döner)
  - Önceki ayarları dönderir

C++

Format Bayraklarını Ayarlama

```
1  // Fig. 11.28: fig11_28.cpp
2  // Demonstrating the flags member fun
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 << "The value of the flags variable is: "
16         << cout.flags()
17         << "\nPrint int and double in original format:\n"
18         << i << '\t' << d << "\n\n";
19    long originalFormat =
20            cout.flags( ios::oct | ios::scientific );
21    cout << "The value of the flags variable is: "
22         << cout.flags()
23         << "\nPrint int and double in a new format\n"
24         << "specified using the flags member function:\n"
25         << i << '\t' << d << "\n\n";
```

```
The value of the flags variable is: 0

Print int and double in original format:
1000    0.0947628

The value of the flags variable is: 4040

Print int and double in a new format
specified using the flags member function:
1750    9.476280e-002
```

C++

Format Bayraklarını Ayarlama

```
26     cout.flags( originalFormat );
27     cout << "The value of the flags variable is:"
28          << cout.flags()
29          << "\nPrint values in original format again:\n"
30          << i << '\t' << d << endl;
31     return 0;
32  }
```

Notice how **originalFormat** (a **long**) is the argument.

```
The value of the flags variable is: 0
Print values in original format again:
1000    0.0947628
```

```
The value of the flags variable is: 0
Print int and double in original format:
1000    0.0947628

The value of the flags variable is: 4040
Print int and double in a new format
specified using the flags member function:
1750    9.476280e-002

The value of the flags variable is: 0
Print values in original format again:
1000    0.0947628
```

C++

Stream Hata Durumları

- **`eofbit`**
  - bir giriş stream'i için end-of-file ile karşılaştıktan sonraki ayarı yapar
  - **`cin'de`** eğer end-of-file ile karşılaşılmışsa **`cin.eof() true`** dönderir

- **`failbit`**
  - Stream için format hatası oluştuğunda ayar yapar
  - eğer stream operasyonu yapılamamışsa **`cin.fail()`** - **`true`** dönderir
  - normalde böyle hataları düzeltme imkanı vardır

C++

Stream Hata Durumları

- **`badbit`**
  - data kaybı ile sonuçlanan hata. Normalde kurtarılamaz
  - eğer stream operasyonu yapılamamışsa **`cin.bad()`** **`true`** dönderir
- **`goodbit`**
  - **`eofbit`**, **`failbit`** veya **`badbit`** 'ten herhangi birinin ayarlı olmadığı durum
  - eğer **`bad`**, **`fail`** ve **`eof`** fonksiyonları **`false`** döndermişse **`cin.good()`** **`true`** dönderir
  - I/O operasyonları sadece "good" streamlerde gerçekleştirilmelidir.
- **`rdstate`**
  - Stream'in durumunu dönderir
  - stream bütün durum bitlerini gözden geçiren **`switch`** ifadesiyle test edilebilir
  - Durumu saptamak için **`eof`**, **`bad`**, **`fail`**, ve **`good`** daha kolaydır

C++

Stream Hata Durumları

- **`clear`**
  - Bir streamin durumunu "good" ' a dönüştürmek için kullanılır
  - **`cin.clear()`** **`cin`**'i temizler ve stream'i **`goodbit`**'e ayarlar.
  - **`cin.clear( ios::failbit )`** **`failbit`**'e ayarlar.

    kullanıcı tanımlı bir türle ilgili problemlerle karşılaşıldığında yapılmalı
- Diğer operatörler
  - **`operator !`**

    eğer **`badbit`** veya **`failbit`** ayarlanmışsa **`true`** dönderir
  - **`operator void*`**

    eğer **`badbit`** veya **`failbit`** ayarlanmışsa **`false`** döndürür
  - Dosya işlemleri için faydalı

C++

Stream Hata Durumları

```
1  // Fig. 11.29: fig11_29.cpp
2  // Testing error states.
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 << "Before a bad input operation:"
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         << "\n\nExpects an integer, but enter a character: ";
19    cin >> x;
20
```

```
Before a bad input operation:
cin.rdstate(): 0
    cin.eof(): 0
   cin.fail(): 0
    cin.bad(): 0
   cin.good(): 1
```

C++

Stream Hata Durumları

```
21    cout << "\nAfter a bad input operation:"
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 << "After cin.clear()"
31         << "\ncin.fail(): " << cin.fail()
32         << "\ncin.good(): " << cin.good() << er
33    return 0;
34 }
```

```
After a bad input operation:
cin.rdstate(): 2
    cin.eof(): 0
   cin.fail(): 1
    cin.bad(): 0
   cin.good(): 0
```

```
Expects an integer, but enter a character: A
```

```
After cin.clear()
cin.fail(): 0
cin.good(): 1
```

C++

Stream Hata Durumları

```
Before a bad input operation:
cin.rdstate(): 0
    cin.eof(): 0
   cin.fail(): 0
    cin.bad(): 0
   cin.good(): 1

Expects an integer, but enter a character: A

After a bad input operation:
cin.rdstate(): 2
    cin.eof(): 0
   cin.fail(): 1
    cin.bad(): 0
   cin.good(): 0

After cin.clear()
cin.fail(): 0
cin.good(): 1
```

C++

Bir Çıkış Stream'ini Bir Giriş Stream'e Bağlama

- **`tie`** üye fonksiyonu
  - **`istream`** ve **`ostream`** operasyonlarının zamanını ayarlar
  - çıktılar sonraki girdilerden önce belirir
  - **`cin`** ve **`cout`** için otomatik olarak yapılır
- **`inputStream.tie(&outputStream);`**
  - **`inputStream`**'i **`outputStream`**'e bağlar
  - **`cin.tie( &cout)`** otomatik olarak yapılır
- **`inputStream.tie( 0 );`**
  - **`inputStream`**'i çıkış streaminden ayırır.

C++

Template (Şablon)

- Daha geniş alanlı ilgili fonksiyonları veya sınıfları kolaylıkla oluşturmaya yarar.
  - fonksiyon Şablon - ilgili fonksiyonların kopyasıdır.
  - Şablon fonksiyon - bir fonksiyon Şablon'dan yapılmış belirli bir fonksiyon.

C++

Fonksiyon Şablon'lar

- Aşırı yüklenmiş fonksiyonlar
  - Değişik türlerden olan verilere benzer işlemler uygularlar
- fonksiyon Şablon'lar
  - değişik türlerden verilere özdeş işlemleri uygularlar
  - Tip denetimi sağlar.
- Biçimi:

  template<class *türü*, **class** *türü*...**>**
  - **class** veya **typename** kullanılabilir- tür paremetrelerini belirtir

  ```
  template< class T >
  template< typename ElementType >
  template< class BorderType, class FillType >
  ```
  - **template** ifadesinden sonra fonksiyon tanımı gelir.

C++

Fonksiyon Şablon'lar

```
1  template< class T >
2  void printArray( const T *array, const int count )
3  {
4     for ( int i = 0; i < count; i++ )
5        cout << array[ i ] << " ";
6
7     cout << endl;
8  }
```

T tür değişkeni. **T**'nin türü bulunuyor ve fonksiyonun içinde yerine konuluyor

Yeni yaratılan fonksiyon derleniyor.

printArray fonksiyonunun **int** kullanılarak yapılan versiyonu

```
void printArray( const int *array, const int count )
{
   for ( int i = 0; i < count; i++ )
      cout << array[ i ] << " ";

   cout << endl;
}
```

C++

Fonksiyon Şablon'lar

```
1  // Fig 12.2: fig12_02.cpp
2  // Using template functions
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  template< class T >
9  void printArray( const T *array, const int count )
10 {
11    for ( int i = 0; i < count; i++ )
12       cout << array[ i ] << " ";
13
14    cout << endl;
15 }
16
17 int main()
18 {
19    const int aCount = 5, bCount = 7, cCount = 6;
20    int a[ aCount ] = { 1, 2, 3, 4, 5 };
21    double b[ bCount ] = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7 };
22    char c[ cCount ] = "HELLO";  // 6th position for null
```

Hangi türden parametre **T'** nin **int**, **float**,vs. yerine kullanıldığına dikkat et .

C++

Fonksiyon Şablon'lar

```
23
24     cout << "Array a contains:" << endl;
25     printArray( a, aCount ); // integer template function
26
27     cout << "Array b contains:" << endl;
28     printArray( b, bCount ); // double template fu
29
30     cout << "Array c contains:" << endl;
31     printArray( c, cCount ); // character template function
32
33     return 0;
34 }
```

Her türden dizi değişik bir türden template fonksiyon tarafından işleme sokuluyor

```
Array a contains:
1 2 3 4 5
Array b contains:
1.1 2.2 3.3 4.4 5.5 6.6 7.7
Array c contains:
H E L L O
```

C++

Şablon Fonksiyonları Aşırı Yükleme

- İlgili Şablon fonksiyonlar aynı isme sahiptirler
  - derleyici aşırı yükleme ayrışma metodlarını kullanarak doğru olanı çağırır
- Fonksiyon Şablon'lar aşırı yüklenebilir
  - diğer fonksiyon Şablon'lar aynı isme sahip olabilir ama farklı sayıda paremetreler içerir
  - Şablon olmayan fonksiyon aynı isimde olabilir ama farklı argümanları vardır
- Derleyici fonksiyon çağırmalarını fonksiyon adı ve argümanları ile eşleştirmeye çalışır
  - Kesin bir eşleşme yoksa, fonksiyon Şablon'lara bakar
  - Bulursa, derleyici Şablon fonksiyonu oluşturur ve kullanır
  - Bulamazsa ya da birden fazla eşleşme bulursa derleyici hata verir

C++

Sınıf Şablon'ları

- Sınıf Şablon'ları
  - Jenerik (generic) sınıfların türe özgü versiyonlarını oluşturmaya yarar
- Biçimi:

```
template <class T>
class SınıfAdı{
  ...
  }
```

  - "**T**" kullanmaya gerek yok, herhangi bir tanıtıcı çalışır
  - Bir sınıf nesnesi oluşturmak için

    SınıfAdı`< tür >` **`myObject;`**

    Örnek: **`Stack< double > doubleStack;`**

C++

Sınıf Şablon'ları

- Şablon sınıf fonksiyonları
  - Normal olarak tanımlanırlar, fakat **`template<class T>`** ile devam eder
  - Sınıftaki jenerik veri **`T`** türü olarak listelenir
  - binary scope resolution (::) operatörü kullanılır
  - Şablon sınıf'a ait fonksiyon tanımı:

```
template<class T>
MyClass< T >::MyClass(int size)
{
  myArray = new T[size];
}
```

  - Yapıcı (constructor) tanımı - **`T`** türünden dizi oluşturur.

C++

Sınıf Şablon'ları

```
1  // Fig. 12.3: tstack1.h
2  // Class template Stack
3  #ifndef TSTACK1_H
4  #define TSTACK1_H
5
6  template< class T >
7  class Stack {
8  public:
9     Stack( int = 10 );     // default constructor (stack size 10)
10    ~Stack() { delete [] stackPtr; } // destructor
11    bool push( const T& ); // push an element onto the stack
12    bool pop( T& );         // pop an element off the stack
13 private:
14    int size;              // # of elements in the stack
15    int top;               // location of the top element
16    T *stackPtr;           // pointer to the stack
17
18    bool isEmpty() const { return top == -1; }      // utility
19    bool isFull() const { return top == size - 1; } // functions
20 };
21
```

C++

Sınıf Şablon'ları

```
22 // Constructor with default size 10
23 template< class T >
24 Stack< T >::Stack( int s )
25 {
26    size = s > 0 ? s : 10;
27    top = -1;                    // Stack is initially empty
28    stackPtr = new T[ size ]; // allocate space for elements
29 }
30
31 // Push an element onto the stack
32 // return 1 if successful, 0 otherwise
33 template< class T >
34 bool Stack< T >::push( const T &pushValue )
35 {
36    if ( !isFull() ) {
37       stackPtr[ ++top ] = pushValue; // place item in Stack
38       return true;  // push successful
39    }
40    return false;      // push unsuccessful
41 }
```

Sınıf template'a ait bir üye fonksiyonun nasıl tanımlandığına dikkat et

Stack dolu mu, boş mu, test et. Dolu değilse eleman ekle.

C++

Sınıf Şablon'ları

```
43 // Pop an element off the stack
44 template< class T >
45 bool Stack< T >::pop( T &popValue )
46 {
47    if ( !isEmpty() ) {
48       popValue = stackPtr[ top-- ];  // remove item from Stack
49       return true;  // pop successful
50    }
51    return false;     // pop unsuccessful
52 }
53
54 #endif
```

Stack dolu mu, boş mu, test et. Eğer boş değilse eleman çıkar

C++

Sınıf Şablon'ları

```
57 #include <iostream>
58
59 using std::cout;
60 using std::cin;
61 using std::endl;
62
63 #include "tstack1.h"
64
65 int main()
66 {
67    Stack< double > doubleStack( 5 );
68    double f = 1.1;
69    cout << "Pushing elements onto doubleStack\n";
70
71    while ( doubleStack.push( f ) ) { // success true returned
72       cout << f << ' ';
73       f += 1.1;
74    }
```

```
Pushing elements onto doubleStack
```

```
1.1 2.2 3.3 4.4 5.5
```

C++

Sınıf Şablon'ları

```
76    cout << "\nStack is full. Cannot push " << f
77         << "\n\nPopping elements from doubleStack\n";
78
79    while ( doubleStack.pop( f ) )  // success true returned
80       cout << f << ' ';
81
82    cout << "\nStack is empty. Cannot pop\n";
83
84    Stack< int > intStack;
85    int i = 1;
86    cout << "\nPushing elements onto intStack\n";
87
88    while ( intStack.push( i ) ) { // success true returned
89       cout << i << ' ';
90       ++i;
91    }
92
```

```
Stack is full. Cannot push 6.6

Popping elements from doubleStack

5.5 4.4 3.3 2.2 1.1

Stack is empty. Cannot pop

Pushing elements onto intStack

1 2 3 4 5 6 7 8 9 10
```

C++

Sınıf Şablon'ları

```
93     cout << "\nStack is full. Cannot push " << i
94          << "\n\nPopping elements from intStack\n";
95
96     while ( intStack.pop( i ) )  // success
97        cout << i << ' ';
98
99     cout << "\nStack is empty. Cannot pop\n";
100    return 0;
101}
```

Stack is full. Cannot push 11

Popping elements from intStack

10 9 8 7 6 5 4 3 2 1

Stack is empty. Cannot pop

```
Pushing elements onto doubleStack
1.1 2.2 3.3 4.4 5.5
Stack is full. Cannot push 6.6

Popping elements from doubleStack
5.5 4.4 3.3 2.2 1.1
Stack is empty. Cannot pop

Pushing elements onto intStack
1 2 3 4 5 6 7 8 9 10
Stack is full. Cannot push 11

Popping elements from intStack
10 9 8 7 6 5 4 3 2 1
Stack is empty. Cannot pop
```

C++

Sınıf Şablon'ları ve Tipsiz Parametreler

- Tipsiz parametreler Şablon'larda kullanılabilir
  - varsayılan argüman
  - **`const`** gibi davranır
- Örnek:

```
template< class T, int elements >
Stack< double, 100 > mostRecentSalesFigures;
```

  - **`Stack< double, 100>`** türünden bir obje tanımlar
  - sınıf tanımında karşılaşılabilir :

```
T stackHolder[ elements ];//stack'i tutmak için dizi
```

  - Çalıştırma zamanındaki dinamik ayırmadan (dynamic allocation) ziyade derleme zamanında diziyi oluşturur,

C++

Sınıf Şablon'ları ve Tipsiz Parametreler

- Sınıflar aşırı-yüklenebilir
  - Şablon sınıf **Array** için **Array<myCreatedType>** adlı bir sınıf tanımla.
  - Böylece bu yeni sınıf **myCreatedType** tipini aşırı-yükler.
  - Şablon aşırı-yüklenmemiş tipler için aynen korunmuş olur.

C++

Şablon'lar ve Miras

- Bir sınıf Şablon, Şablon bir sınıftan türetilebilir.
- Bir sınıf Şablon, Şablon olmayan bir sınıftan türetilebilir.
- Bir Şablon sınıf, bir sınıf Şablon'dan türetilebilir.
- Bir Şablon olmayan sınıf, bir sınıf Şablon'dan türetilebilir.

C++

Şablon'lar ve Arkadaşlık (friendship)

- Bir sınıf Şablon ve aşağıdakiler arasında arkadaşlığa izin vardır
  - global fonksiyon
  - başka bir sınıfın üye fonksiyonları
  - tüm sınıf
- Arkadaş (friend) fonksiyonlar
  - sınıf Şablon **`X`**'in tanımı içinde :
  - **`friend void f1();`**
    - **`f1()`** bütün Şablon sınıfın arkadaşı
  - **`friend void f2( X< T > & );`**
    - **`f2( X< int > & )`**, **`sadece X< int >`** 'in arkadaşı . Aynı şey **`float`**, **`double`**, vs. için de geçerli

C++

Şablon'lar ve Arkadaşlık (friendship)

- `friend void A::f3();`
  - Sınıf `A`'nın üye fonksiyonu `f3` bütün Şablon sınıfların arkadaşıdır
- `friend void C< T >::f4( X< T > & );`
  - `C<float>::f4( X< float> & )` sadece `class X<float>` sınıfının arkadaşıdır

- **`Arkadaş(Friend)`** sınıflar
  - `friend class Y;`
    - `Y`'nin her üye fonksiyonu `X`'ten yapılan her Şablon sınıfın arkadaşıdır
  - `friend class Z<T>;`
    - class `Z<float>` , class `X<float>` 'nin arkadaşıdır, vs.

C++

Şablon'lar ve Statik Üyeler

- Şablon olmayan sınıflar
  - **`static`** üye veriler bütün nesneler tarafından paylaşılır
- Şablon sınıflar
  - her sınıfın (**`int`**, **`float`**, etc.) kendi **`static`** üye verilerinin kopyası vardır
  - **`static`** değişkenlere dosya kapsamında ilk değer ataması yapılır
  - her Şablon sınıfın kendi **`static`** üye fonksiyonlarının kopyası vardır.