C++ Dersi:
Nesne Tabanlı Programlama

2. Baskı

Bölüm 19: Standart Şablon Kütüphanesi (vector)

Çiğdem Turhan

Fatma Cemile Serçe

İçerik

Hedefler

- Standart şablon kütüphanesinin yapısını ve bileşenlerini anlatma
- vector nesnesi ve diziler arasındaki farkları anlatma
- vector tanımı yapma
- vector nesnesinin elemanlarına erişme
- vector nesnesinin sonuna ekleme
- vector nesnesinin ilk ve son elemanına erişme
- vector nesnesinin büyüklüğünü hesaplama
- vector nesneleri arasında eleman değiş tokuşu yapma
- vector nesnesinin elemanları üzerinde işlem yapmak için iteratör kullanma

19.1 Standart Şablon Kütüphanesi

- STL kütüphaneleri üç bileşenden oluşur:
  - ***konteyner*** (container): veri gruplarını saklamaya yarayan veri yapısı.
  - ***iteratör*** (iterator): konteynerde saklanan verilere erişmek için tanımlanmış genel gösterge.
  - ***algoritma*** (algorithm): konteyner nesneleri üzerinde işlem yapabilen genel fonksiyon şablonları
- STL'de yer alan konteynerler de iki kategoriye ayrılır:
- ***sıralı konteyner*** (sequence container): dizilerde olduğu gibi veriler ardarda saklanır.
- ***ilişkisel konteyner*** (associative container): veriler anahtarıyla beraber saklanır ve bu sayede verilere hızlı ve rastgele erişim yapılabilir.

19.2 vector Sınıfı

- vector konteyneri
  - istenildiğinde genişletilebilen,
  - içinde birçok üye fonksiyon barındıran,
  - dinamik bir dizi şablonu
- Dizilerde olduğu gibi bir vector nesnesi bellekte elemanlarını ardarda saklar ve eleman erişimini [ ] operatörü ile yapabilir.
- Ancak vector konteynerlerinin dizilere oranla birçok avantajı vardır.
  - Dizi tanımının aksine vektor nesnesi tanımında kaç elemanlı olacağını belirtmeye gerek yoktur.
  - vector nesnesi yeni eleman eklendiğinde otomatik olarak genişler.
  - vector nesnesi içinde kaç eleman içerdiğini tutar.

19.3 vector Tanımı

- **#include** <vector>
- vector konteynerinin içinde farklı yapıcı fonksiyonlar tanımlanmıştır:
  - **vector<int> v1;** : v1 isimli, 0 elemanlı, int tipinde bir vektör tanımlar.
  - **vector<double> v2(10);** : v2 isimli, double tipinde 10 elemanlı bir vektör tanımlar.
  - **vector<int> v3(10,4);** : v3 isimli, int tipinde 10 elemanlı bir vektör tanımlar. Her elemanın ilk değeri 4 olarak atanır.
  - **vector<int> v4(v3);** : v4 isimli vektör tanımlar ve içine v3 vektörünün içeriğini kopyalar.

19.4 `vector` Elemanlarına Erişim

- `[]` operatörü veya `at()` fonksiyonu
- **`v.at(indeks)`** : v vektörünün indeks pozisyonundaki elemanına erişir. Bu fonksiyon geçerli indeksin girilip girilmediğini kontrol eder.
- **`v[indeks]`:** v vektörünün indeks pozisyonundaki elemanına erişir. Bu fonksiyon geçerli indeksin girilip girilmediğini kontrol etmez.

19.4 vector Elemanlarına Erişim...

Örnek 19.1

```
#include <iostream>
#include <vector>
using namespace std;
int main (void)
{
    vector<int> v(5,2);          // 5 elemanlı içine 2 atanmış vektör tanımı
    v[0]+=1;
    v[1]+=2;
    for (int i=0;i<5;i++)        // v vektörünün  elemanları yazdırılır
       cout<<" v["<<i<<"]="<<v.at(i)<<endl;
    return 0;
}
```

Çıktı

```
v[0]=3
v[1]=4
v[2]=2
v[3]=2
v[4]=2
```

19.5 vector Üye Fonksiyonlar

- **push_back(y)**: Vektör'ün sonuna y elemanını ekler.
- **front() veya v[0]**: Vektör'ün ilk elemanını döndürür.
- **back()**: Vektör'ün son elemanını döner.

Örnek 19.2

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> a, b;          // a ve b isimli iki vektör tanımlanır
    a.push_back(10);           // a'ya 10 ve 20, b'ye 30 eklenir
    a.push_back(20);
    b.push_back(30);
    cout<<a.front()<<endl;     // a'nın ilk elemanı yazdırılır
    cout<<a.back()<<endl;      // a'nın son elemanı yazdırılır
    cout<<b[0]<<endl;          // b'nin ilk elemanı yazdırılır
    return 0;
}
```

Çıktı

```
10
20
30
```

19.5 vector Üye Fonksiyonlar...

- **pop_back()** : Vektör'deki son elemanı siler.

Örnek 19.3

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> a;                  // a isimli  vektör tanımlanır
    a.push_back(10);                // a'ya 10, 20 ve 30 eklenir
    a.push_back(20);
    a.push_back(30);
    a.pop_back();                   // a'nın son iki elemanını siler
    a.pop_back();
    cout<<a.back()<<endl;           // a'nın son elemanı yazdırılır
    return 0;
}
```

19.5 vector Üye Fonksiyonlar...

Çıktı

```
10
```

19.5 vector Üye Fonksiyonlar...

- **size()** : Vektör'de tutulan eleman sayısını döner.

Örnek 19.4

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> a;                  // a isimli  vektör tanımlanır
    a.push_back(10);                // a'ya 10, 20 ve 30 eklenir
    a.push_back(20);
    a.push_back(30);
    cout<<a.size()<<endl;           // a'nın eleman sayısını yazdırır
    return 0;
}
```

Çıktı

```
3
```

19.5 vector Üye Fonksiyonlar...

- **empty()** : Vektör'ün boş olup olmama durumuna göre true ya da false değer döner.

Örnek 19.5

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> a;                  // a isimli  vektör tanımlanır
    a.push_back(10);                // a'ya 10, 20 ve 30 eklenir
    a.push_back(20);
    a.push_back(30);
    cout<<a.size()<<endl;           // a'nın eleman sayısını yazdırır
    while(!a.empty()){              // a boş oluncaya kadar döner
        a.pop_back();               // a'dan bir eleman siler
    }
    cout<<a.size()<<endl;           // a'nın eleman sayısını yazdırır
    return 0;
}
```

Çıktı

```
3
0
```

19.5 vector Üye Fonksiyonlar...

- **swap(v2)** : Vektör'ün elemanları ile v2 vektörünün elemanları değiş tokuş edilir.

Örnek 19.6

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> a;                  // a isimli  vektör tanımlanır
    vector<int> b;                  // b isimli  vektör tanımlanır
    a.push_back(10);                // a'ya 10, 20, b'ye 30, 40 eklenir
    a.push_back(20);
    b.push_back(30);
    b.push_back(40);
    a.swap(b);                      // a ve b değiş tokuş edilir
    cout<<a.front()<<endl;          // a'nın ilk elemanı yazdırılır
    cout<<b.front()<<endl;          // b'nin ilk elemanı yazdırılır
    return 0;
}
```

Çıktı

```
30
10
```

19.5 vector Üye Fonksiyonlar...

- **clear()** : Vektör'ün içerisindeki tüm elemanları siler.

Örnek 19.9

```
#include <iostream>
#include <vector>
using namespace std;
int main (void)
{
    vector<int> a;                  // a isimli  vektör tanımlanır
    a.push_back(10);                // a'ya 10, 20, 30  eklenir
    a.push_back(20);
    a.push_back(30);
    a.clear();                      // a'nın içeriği silinir
    cout<<a.size()<<endl;           // a'nın eleman sayısını yazdırır
    return 0;

}
```

Çıktı

```
0
```

19.6 İteratörler

- *Ing. Iterator*
- Bir veri yapısı içerisinde tutulan elemanlar üzerinde işlem yapmak amacı ile kullanılan bir göstergedir.
- Vektörlere ait iteratörleri kullanmak için aşağıdaki söz dizimine uygun tanımlama yapmak gerekir.

```
vector<VeriTipi>::iterator iteratörAdı;
```

- Vektör gibi içerisinde veri saklayan her konteynerin, `begin()` ve `end()` fonksiyonları kullanılarak iteratörlerle erişilir.

19.6 İteratörler...

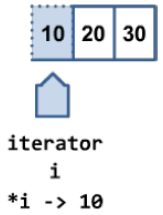
- **`begin()`** : Vektör'ün ilk elemanını gösteren iteratörü döndürür.

Örnek 19.10

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> a;                  // a isimli  vektör tanımlanır
    vector<int>::iterator i;        // i isimli iteratör tanımlanır
    a.push_back(10);                // a'ya 10, 20, 30  eklenir
    a.push_back(20);
    a.push_back(30);
    i = a.begin();                  // i iteratörü a'nın ilk elemanını gösterir
    cout<<*i<<endl;                 // i'in gösterdiği ilk eleman yazdırılır
    return 0;
}
```

Çıktı

```
10
```



Şekil 19.1 Vektör Nesnesinin İlk Elemanını Gösteren İteratör

19.6 İteratörler...

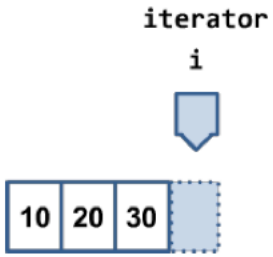
- **`end()`** : Vektörün son elemanından sonrasını gösteren iteratörü döndürür.

Örnek 19.11

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> a;                  // a isimli  vektör tanımlanır
    vector<int>::iterator i;        // i  isimli iteratör tanımlanır
    a.push_back(10);                // a'ya 10, 20, 30  eklenir
    a.push_back(20);
    a.push_back(30);
    i = a.end();                    // i a'nın son elemanından sonrasını gösterir
    cout<<*i<<endl;                 // i'nin gösterdiği hücredeki veri  yazdırılır
    return 0;
}
```

Çıktı

```
1414422387
```



Şekil 19.2 Vektörün Son Elemanından Sonrasını Gösteren İteratör

19.6 İteratörler...

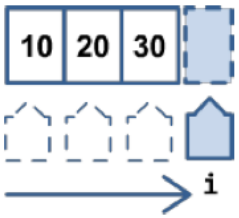
- **++** : İteratör veri yapısındaki elemanlar üzerinde bir kademe ilerler ve bir sonraki elemanı gösterir.

Örnek 19.12

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> a;                  // a isimli  vektör tanımlanır
    vector<int>::iterator i;        // i  isimli iteratör tanımlanır
    a.push_back(10);                // a'ya 10, 20, 30  eklenir
    a.push_back(20);
    a.push_back(30);
    for(i=a.begin();i!=a.end();++i){  // a'nın başından sonuna kadar döner
        cout<<*i<<endl;                // i'nin şu anda gösterdiği eleman yazdırılır
    }
    return 0;
}
```

Çıktı

```
10
20
30
```



Şekil 19.3 Vektörün İlk Elemanından Başlayarak Gezinme

19.6 İteratörler...

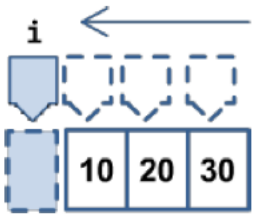
- `--` : İteratör veri yapısındaki elemanlar üzerinde bir kademe geri gelir ve bir önceki elemanı gösterir.

Örnek 19.13

```
#include <iostream>
#include <vector>
using namespace std;
int main (void)
{
    vector<int> a;                  // a isimli  vektör tanımlanır
    vector<int>::iterator i;        // i  isimli iteratör tanımlanır
    a.push_back(10);                // a'ya 10, 20, 30  eklenir
    a.push_back(20);
    a.push_back(30);
    i = a.end();                    // i iteratörü a'nın son elemanından sonrasını gösterir
    for(--i;i!=a.begin();--i){      // a'nın sonundan başına kadar döner
        cout<<*i<<endl;             // i'nin şu anda gösterdiği eleman yazdırılır
    }
    cout<<*i<<endl;
    return 0;
}
```

Çıktı

```
30
20
10
```



Şekil 19.4 Vektörün Son Elemanından Geriye Doğru Gezinme