C++ Dersi:
Nesne Tabanlı Programlama

2. Baskı

Bölüm 16: Kural Dışı Durum Yönetimi

Çiğdem Turhan

Fatma Cemile Serçe

İçerik

16.1 Kural Dışı Durum Yönetim Komutları

- **16.1.1** `try` Komutu
- **16.1.2** `throw` Komutu
- **16.1.3** `catch` Komutu

16.2 Fonksiyon Tanımlarında `throw` Kullanımı

16.3 Çoklu Kural Dışı Durum Yönetimi

16.4 `catch(...)` Bloğu

16.5 Hatayı Yeniden Fırlatma

16.6 Sınıflarla Hata Yakalama

Çözümlü Sorular

Hedefler

- Kural dışı durumlara örnekler verme
- Verilen bir kural dışı durumun yönetimi için `try-catch` komutlarını kullanma
- Fonksiyon tanımlarındaki `throw` komutunu dikkate alarak uygun `try-catch` blokları yazma
- `try-catch` bloğu kullanıldığında ve kural dışı durum oluştuğunda programın nasıl davranacağını anlatma
- Program akışında birden çok kural dışı durum olma olasılığını düşünerek uygun catch tanımlamaları yapma
- Program akışında oluşabilecek her türlü hatayı yakalayan `catch` blok tanımı yapma
- Atılan hatayı yakalayıp tekrar fırlatan catch bloğunu yazma
- Verilen tanımlamalara uygun olarak, fırlatılacak hatayı tanımlayan sınıf tanımı yapma

Kural Dışı Durum

- *İng. Exception*
- Programın karşılaştığı beklenmedik durum
- Bir programcının temel amacı, yazdığı programın her koşulda çalışmasını sağlamaktır.
- Programcının karşılaşılabilecek her beklenmeyen durumu öngörerek, programını hata vermeden çalışacak şekilde tasarlaması gerekir.
- Programın, çalışması sırasında oluşan hataları veya kural dışı durumları yakalayıp, önlem alması ve çalışmaya devam etmesi hedeflenir.

16.1 Kural Dışı Durum Yönetim Komutları

- C++ programlama dilinde kural dışı durum yönetimi üç farklı komut ile gerçekleştirilir.
  - `try` (deneme)
  - `throw` (fırlatma)
  - `catch` (yakalama)
- Program akışında programcı tarafından öngörülmeyen kural dışı bir durum oluşursa, program tamamen durur ve/veya derleyici hata verir. Kural dışı durum yönetimi ile ise programın durmasına gerek kalmadan, oluşan hatalar çözümlenir

16.1 Kural Dışı Durum Yönetim Komutları

Örnek 16.1

```
#include <iostream>
using namespace std;
int main()
{   int sayi;
    try{
        cout<<"Pozitif bir sayi giriniz:";
        cin>>sayi;
        if (sayi<0)                 // Girilen sayı negatif ise hata fırlatılır
            throw "Negatif sayi girilmistir";
        cout<<sqrt(sayi)<<endl; // Girilen sayı pozitif ise karakökü yazılır
    }catch (const char* str){    //Fırlatılan hatanın mesajını yazdıran catch bloğu
        cout<<str<<endl;
    }
    cout<<"Calismaya devam eder..."<<endl;
    return 0;
}
```

Çıktı 1

```
Pozitif bir sayi giriniz:64
8
Calismaya devam eder...
```

Çıktı 2

```
Pozitif bir sayi giriniz:-12
Negatif sayi girilmistir
Calismaya devam eder...
```

16.1.1 try Komutu

- try anahtar kelimesi deneme anlamına gelmektedir.
- Burada kastedilen try bloğu, içerisinde yer alan komutların çalıştırılmasının denenmesidir.
- Sözdizimi:

```
try{
        // komutlar
}
```

```
try{
    cout<<"Pozitif bir sayi giriniz:";
    cin>>sayi;
    if (sayi<0)
        throw "Negatif sayi girilmiştir";
    cout<<sqrt(sayi)<<endl;
}
```

Burada try bloğu içerisinde kullanıcıdan alınan sayının karekökünün hesaplanması denenir.
Eğer kullanıcı geçerli bir sayı girerse bu deneme başarı ile sonuçlanır.
Ancak geçersiz (negatif) bir sayı girildiğinde ise kural dışı durum oluşur.

16.1.2 throw Komutu

- `throw` komutu, kural dışı durum oluştuğu anda kullanılır.
- `throw` komutunda belirtilen parametre ile hata fırlatılır.
- `try-catch` bloğu içerisinde fırlatılan bu hata, uygun `catch` bloğu bulunursa yakalama işlemi gerçekleşir.
- Sözdizimi:

```
throw  <parametre>;
```

- Örnek:

```
throw "Negatif sayi girilmiştir";
throw 4;
```

16.1.2 throw Komutu...

- Bazı kural dışı durumlarda programın çalışmasını durdurmak isteyebiliriz. Bu durumlarda throw komutu try-catch bloğu olmadan kullanılır.
- Örnek:
  - Bu örnekte 10-elemanlık dinamik bir dizi için bellekten yer alınması gerekmektedir. Eğer söz konusu dizi için gerekli yer alınabilirse "Bellekten yer basariyla alindi..." mesajı çıktı olarak verilecektir. Eğer dinamik yer alımı sürecinde bir problem oluşursa programın devam etmesi anlamsız olur ve throw komutu ile program durdurulur.

Örnek 16.2

```
#include <iostream>
using namespace std;
int main()
{   int* say = new int[10];
    if (say==0){     // new komutu için bellekten yer alınamadıysa hata fırlatılır
        throw "Bellekten yer alinamadi!!!";
    }else            // new komutu için bellekten yer alındıysa mesaj yazdırılır
        cout<<" Bellekten yer basariyla alindi...";
    say[0]=1;
    say[5]=-3;
    return 0;
}
```

Çıktı

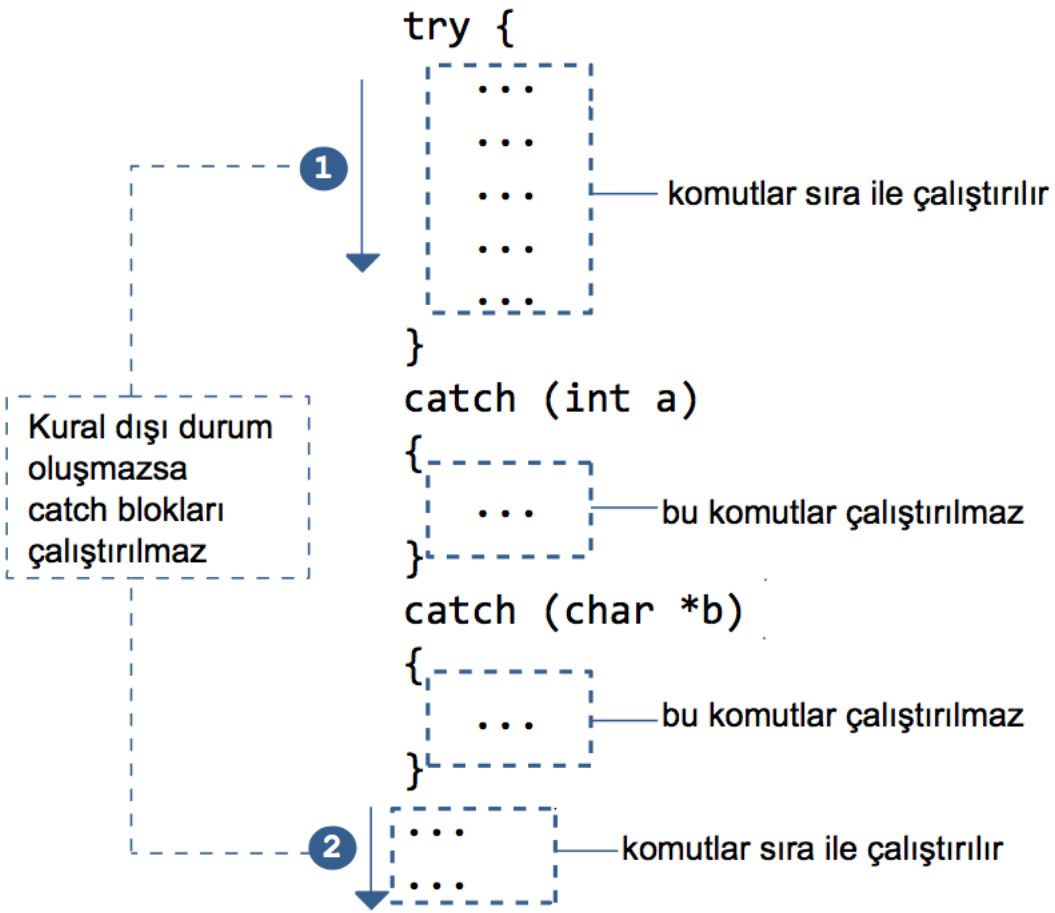
```
Bellekten yer basariyla alindi...
```

16.1.3 catch Komutu

- try bloğu içerisinde yer alan komutların çalıştırılması sırasında fırlatılan hataların yakalanması için kullanılır.
- Her catch bloğu mutlaka bir try bloğu ile ilişkilendirilmelidir.
- catch bloğunu yazarken try bloğunun hemen ardından gelmesine ve araya başka komutların yazılmamasına dikkat etmemiz gerekir.
- Sözdizimi:

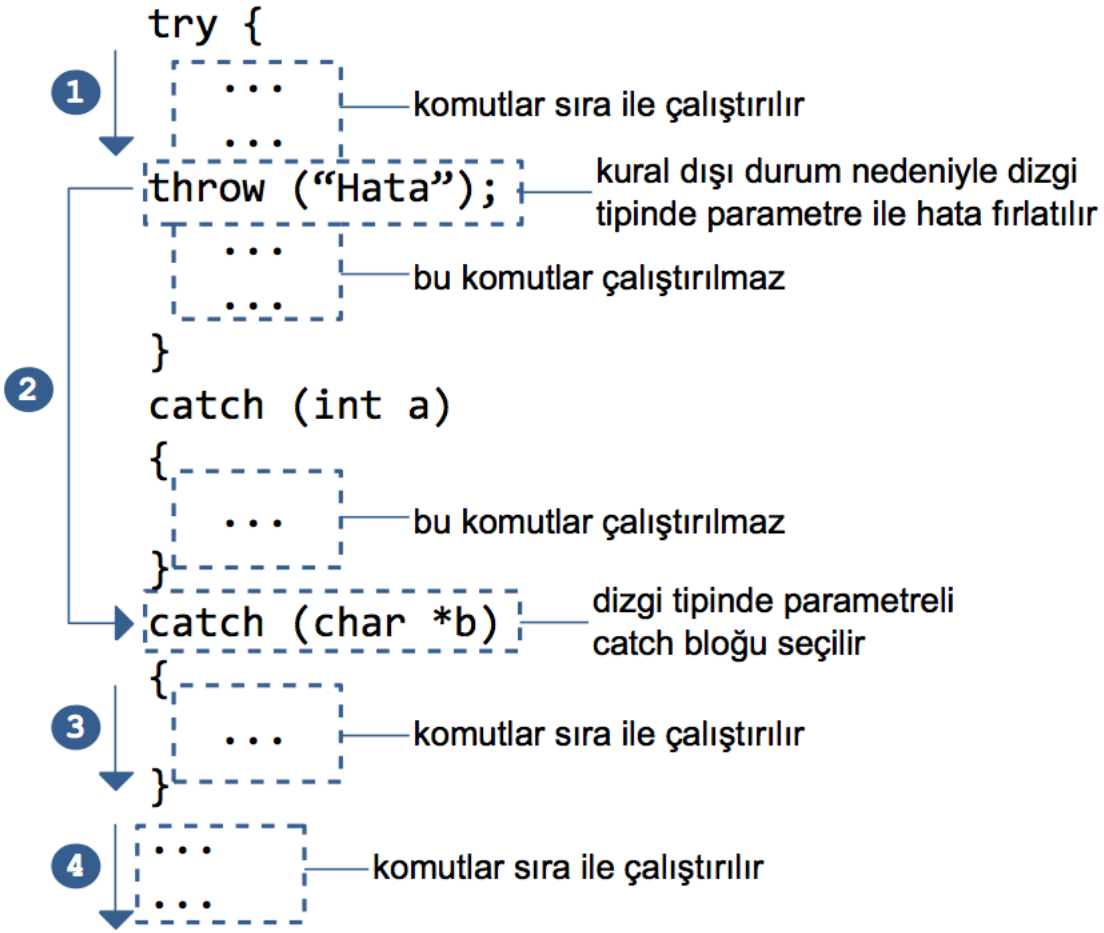
```
try{
    // Komutlar
    // Kural dışı durum oluşursa
        throw (Parametre)
    // Komutlar
}catch(Parametre){
    // Kural dışı durum oluştuğunda çalıştırılacak komutlar
}
```

16.1.3 catch Komutu...



Şekil 16. 1 Kural Dışı Durum Olmaması Durumunda `try-catch` Bloğunun Akış Şeması

16.1.3 catch Komutu...



Şekil 16. 2 Kural Dışı Durum Oluşması Durumunda `try-catch` Bloğunun Akış Şeması

16.1.3 catch Komutu...

Örnek 16.3

```
#include <iostream>
using namespace std;
int main()
{
    int sayi[] = {1,-3,5,-7,9,-11,13,-15,17,-19};
    int indeks;
    cout<<"Indeks değeri giriniz:";
    cin>>indeks;
    cout<<"Sayi:"<<sayi[indeks]<<endl; // sayi dizininin indeks. elemanını yazdırır
    cout<<"Calismaya devam eder...";
    return 0;
}
```

Çıktı 1

```
Indeks degerini giriniz:1
Sayi:-3
Calismaya devam eder...
```

Çıktı 2

```
Indeks degeri giriniz:-22
Sayi:2203496
Calismaya devam eder...
```

16.1.3 catch Komutu...

Örnek 16.4

```
#include <iostream>
using namespace std;
int main()
{
    int sayi[] = {1,-3,5,-7,9,-11,13,-15,17,-19};
    int indeks;
    try{
       cout<<"Indeks degeri giriniz:";
       cin>>indeks;
       if ((indeks>=10)||(indeks<0))//Girilen indeks dizi indeksleriyle kontrol edilir
         throw indeks;                    // Girilen indeks geçersizse hata fırlatılır
       //Indeks geçerliyse indeks. eleman yazdırılır
       cout<<"Sayi:"<<sayi[indeks]<<endl;
    }catch (const int indeks) {  // Hatalı indeksi yazdıran catch bloğu
        cout<<"Gecersiz Indeks:"<<indeks<<endl;
    }
    cout<<"Calismaya devam eder...";
    cout<<endl;
    return 0;
}
```

Çıktı

```
Indeks degeri giriniz:-22
Gecersiz Indeks:-22
Calismaya devam eder...
```

16.2 Fonksiyon Tanımlarında `throw` Kullanımı

- `throw` komutu bir fonksiyon başlığının sonunda da kullanılabilir.
- Bu kullanım ile, söz konusu fonksiyonda kural dışı bir durum oluşabileceği ifade edilmiş olur.
- Sözdizimi:

VeriTipi fonksiyonAdı(parametre listesi) **`throw`** *(parametre tipi)*
{…}

16.2 Fonksiyon Tanımlarında throw Kullanımı...

Örnek 16.5

```
#include <iostream>
using namespace std;
// throw kullanan, indeks. elemanı döndüren fonksiyon
int sayiAl(int indeks) throw (int)
{
    int sayi[] = {1,-3,5,-7,9,-11,13,-15,17,-19};
    if ((indeks>=10)||(indeks <0))// Girilen indeks dizi indeksleriyle kontrol edilir
        throw indeks;              // Girilen indeks geçersizse hata fırlatılır
    return sayi[indeks];           // Indeks geçerliyse indeks. eleman döndürülür
}
int main()
{
    int indeks;
    cout<<"Indeks degeri giriniz:";
    cin>>indeks;
    try{                                  // Komutlar denenir
        cout<<sayiAl(indeks);
    }catch(int i){                        // Hatalı indeksi yazdıran catch bloğu
        cout<<"Gecersiz indeks degeri:"<<i<<endl;
    }
    cout<<"Calismaya devam eder...";
    return 0;
}
```

Çıktı

```
Indeks degeri giriniz:-22
Gecersiz indeks degeri:-22
Calismaya devam eder...
```

16.3 Çoklu Kural Dışı Durum Yönetimi

- Bir try bloğunda birden çok komut yer alabilir ve bu komutlardan bir ya da daha fazlası farklı kural dışı durumlara neden olabilir.
- Her kural dışı durum için ayrı ayrı catch bloğu tanımı yapılmalıdır.

16.3 Çoklu Kural Dışı Durum Yönetimi

Örnek 16.6

```
#include <iostream>
#include "math.h"
using namespace std;
int sayiAl(int indeks) throw (int)              // indeks. elemanı döndürür
{   int sayi[] = {1,-3,5,-7,9,-11,13,-15,17,-19};
    if ((indeks>=10)||(indeks <0))
        throw indeks;
    return sayi[indeks];
}
float karekokAl(int sayi) throw(const char*) // Sayının karekökünü döndürür
{   if (sayi<0)
        throw "Negatif Sayi";
    return sqrt(sayi);
}
int main()
{   int indeks;
    cout<<"Indeks degeri giriniz:";
    cin>>indeks;
    try{
        int sayi = sayiAl(indeks);
        cout<<"Sayi:"<<sayi<<endl;
        cout<<"Karekok:"<<karekokAl(sayi)<<endl;
    }catch(const int i){                        // Indeks hatası
        cout<<"Gecersiz indeks degeri:"<<i<<endl;
    }catch(const char* hata){                   // Negatif sayı hatası
        cout<<hata<<endl;
    }
    cout<<"Calismaya devam eder..."<<endl;
    return 0;
}
```

Çıktı 1

```
Indeks degeri giriniz:2
Sayi:5
Karekok:2.23607
Calismaya devam eder...
```

Çıktı 2

```
Indeks degeri giriniz:-1
Gecersiz indeks degeri:-1
Calismaya devam eder...
```

Çıktı 3

```
Indeks degeri giriniz:3
Sayi:-7
Negatif Sayi
Calismaya devam eder...
```

16.4 `catch(...)` Bloğu

- Bir catch bloğunda parametre tanımı yerine “…” (üç nokta) kullanıldığında fırlatılabilecek her kural dışı durum yakalanır.

16.4 catch(...) Bloğu...

Örnek 16.7

```
#include <iostream>
#include "math.h"
using namespace std;
int sayiAl(int indeks) throw (int)              // Indeks. elemanı döndürür
{
    int sayi[] = {1,-3,5,-7,9,-11,13,-15,17,-19};
    if ((indeks>=10)||(indeks <0))
        throw indeks;
    return sayi[indeks];
}
float karekokAl(int sayi) throw(const char*) // Sayının karekökünü döndürür
{
    if (sayi<0)
        throw "Negatif Sayi";
    return sqrt(sayi);
}
int main()
{
    int indeks;
    cout<<"Indeks degeri giriniz:";
    cin>>indeks;
    try{
        int sayi = sayiAl(indeks);
        cout<<"Sayi:"<<sayi<<endl;
        cout<<"Karekok:"<<karekokAl(sayi)<<endl;
    }catch(...){                                // Tüm hataları yakalar
        cout<<"Hata Olustu."<<endl;
    }
    cout<<"Calismaya devam eder..."<<endl;
    return 0;
}
```

Çıktı 1

```
Indeks degeri giriniz:0
Sayi:1
Karekok:1
Calismaya devam eder...
```

Çıktı 2

```
Indeks degeri giriniz:-10
Hata Olustu.
Calismaya devam eder...
```

Çıktı 3

```
Indeks degeri giriniz:1
Sayi:-3
Hata Olustu.
Calismaya devam eder...
```

16.5 Hatayı Yeniden Fırlatma

- Program akışı içerisinde bir hata yakalanmış, gerekli işlemler yapılmış ve aynı hata yeniden fırlatılarak programın durdurulması istenenebilir.
- Bu durumda `throw` komutu aşağıdaki söz dizimi ile kullanılabilir.

```
throw;
```

16.5 Hatayı Yeniden Fırlatma...

Örnek 16.8

```
#include <iostream>
#include "math.h"
using namespace std;
int sayiAl(int indeks) throw (int)              // Indeks. elemanı döndürür
{
    int sayi[] = {1,-3,5,-7,9,-11,13,-15,17,-19};
    if ((indeks>=10)||(indeks <0))
        throw indeks;
    return sayi[indeks];
}
float karekokAl(int sayi) throw(const char*) // Sayının karekökünü döndürür
{
    if (sayi<0)
        throw "Negatif Sayi";
    return sqrt(sayi);
}
```

```
int main()
{
    int indeks;
    cout<<"Indeks degeri giriniz:";
    cin>>indeks;
    try{
        int sayi = sayiAl(indeks);
        cout<<"Sayi:"<<sayi<<endl;
        cout<<"Karekok:"<<karekokAl(sayi)<<endl;
    }catch(...){
        cout<<"Hata Olustu!"<<endl;
        throw;                                  // Hata tekrar fırlatılır
    }
    cout<<"Calismaya devam eder..."<<endl;
    return 0;
}
```

Çıktı

```
Indeks degeri giriniz:1
Sayi:-3
Hata Olustu!
```