C++ Programlama Dilinde Hata Yakalama

Hata yakalamadan, kasıt çalışma anında beklenmeyen durumların oluşumunu yakalayabilmektir. Bu tür hatalar yakalanmadığı zaman programın çökmesine neden olabilir. Hata oluşumunda bu hatanın fırlatılması ve try catch blokları yardımıyla fırlatılan hataların yakalanması gerekir. C++ programlama diline bakıldığında standart kütüphanelerin birçoğu hata fırlatmaz. Dolayısıyla hata fırlatılmadığı için yakalanamayacaktır. Bundan dolayı C++ programlama dilinde hata sınıflarını programcıların kendileri yazması gerekmektedir. Veri yapıları dersi kapsamında ileri ki konularda sıkça karşınıza çıkacak olan bu hata yakalama konusunu iyi anlamak gerekmektedir. Örneğin bir yığıtta eleman kalmadığı zaman pop mesajının çağrılması hata mesajının fırlatılmasını gerektirmektedir. Adım adım bunun nasıl yapılacağı aşağıda verilmiştir. Önce bir hata sınıf yazılır.

```
class Hata{
   private:
      string mesaj;
   public:
      Hata(const string &msg):mesaj(msg){ }
      string Mesaj(){
         return mesaj;
      }
};
```

Yukarıdaki hata sınıf incelendiğinde, alt alan olarak hatanın ne olduğunu tutan hata mesajı ve bu mesaja erişebilmek için Mesaj fonksiyonu bulunmaktadır.

Çok sık karşılaşılabilen sıfıra bölünme hatasını göstermek için yukarıda yazılmış olan Hata sınıfından kalıtlayarak yeni bir sınıf tanımlanır.

```
class SifiraBolunme : public Hata{
  public:
      SifiraBolunme(const string &msg):Hata(msg) {
     }
};
```

Sıfıra bölünme sınıfının yaptığı tek şey yapıcı metodunda almış olduğu mesajı Hata sınıfına iletmektir. Bu hata fırlatma aşağıdaki örnekte nasıl kullanıldığı gösterilmektedir.

```
#include "SifiraBolunme.hpp"
int main()
{
  try
  {
    double x,y;
    cin>>x;
    cin>>y;
    if(y==0)throw SifiraBolunme("Sıfıra bölünme hatası");
    cout<<"x/y="<<x/y;
  }
  catch(SifiraBolunme &sfr)
  {
    cout<<sfr.Mesaj()<<endl;
  }
  return 0;
}
```

Yukarıdaki örnekte y değeri sıfır geldiği anda hata fırlatılmaz ise sıfıra bölünme hatasından dolayı program çökecektir. Ama yukarıdaki örnekteki gibi hata fırlatıldığı için sıfıra bölünme hata mesajı ekrana gelecektir.

Örneğin aşağıdaki diğer bir örnekte, Sayidizisi sınıfı tanımlanmış bu 3 adet sayıyı dizide tutan bir sınıf. Sahip olduğu getir fonksiyonu sayesinde parametre ile verilen indeksteki değeri döndürüyor. Fakat burada dizi boyutundan daha büyük veya negatif bir sayı girildiğinde dizi sınırları dışında şeklinde bir hata fırlatılması gerekiyor. İşte bunun için Tasma isimli bir sınıf yine hata sınıfından kalıtlıyor. Hata fırlatma işlemi doğal olarak Getir metodunun içinde yapılıyor.

```
class Tasma : public Hata{
	public:
		Tasma(const string &msg):Hata(msg){
		}
};
```

```
class SayiDizisi{
	private:
		int x[3];
	public:
		SayiDizisi(){
			x[0]=0;
			x[1]=0;
			x[2]=0;
		}
		SayiDizisi(int x1,int x2,int x3){
			x[0]=x1;
			x[1]=x2;
			x[2]=x3;
		}
		int Getir(int index){
			if(index>2 || index<0) throw Tasma("Dizi sınırları dışında.");
			return x[index];
		}
};
```

```
int main()
{
 SayiDizisi *dizi = new SayiDizisi(24,85,120);
 try
 {
  int index;
  cout<<"index: ";
	cin>>index;
	cout<<dizi->Getir(index);
 }
 catch(Tasma &tsm)
 {
  cout<<tsm.Mesaj()<<endl;
 }
```

```
  delete dizi;
  return 0;
}
```

Hazırlayan
Arş. Gör. M. Fatih ADAK