

## \* Exception Handling:

- Forcing mekanizması var. Ya handle edilir, ya da sonlanır.
- Happy path ve unhappy path birbirinden temiz bir şekilde ayrılmış.
- Hatayı isteyen kod ile, hatayı handle eden kod arasında doğrudan bir link vardır.

### - Exception Throwing:

- Fonksiyon işini yapamadığında bir nesne oluşturur. Bu nesnenin varlık nedeni, işini yapamadığını okuyana aktarmak ve kodu araştırmaktır.
- Bu nesneyi yakalayan kod müdahale eder.
- Gönderilen hata ya yakalanmalı, ya da kaynak kaybı yaşanmadan program sonlandırılmalı. (resumptive / terminative)

### - Try Block:

- Try Block da blockscope kurallarına tabi. (otomatik olarak nesneler, block sonu hata ile biter)

### - Catch:

```
try {  
    ///  
}  
catch(int) {  
    ///  
}  
catch(double) {  
    ///  
}  
catch(unsigned int) {  
    ///  
}
```

Gönderilen hata nesneleri ilgili catch handler'a girer

→ ift parametre ile catch yapılır.

→ elipsis (...) ile type gösterilerek catch edilir.

### - Set\_terminate function:

```
// using terminate_handler = void (*)(void);  
typedef void (*terminate_handler)();
```

```
terminate_handler set_terminate(terminate_handler);
```

```
void necfunc()  
{  
    std::cout << "necfunc çağrıldı çünkü necfunc işlevini std::terminate çağırdı\n";  
    std::exit(EXIT_FAILURE);  
}  
  
int main()  
{  
    set_terminate(&necfunc);  
    std::cout << "main çağrıldı\n";  
    f1();  
    std::cout << "main sona erdi\n";  
}
```

bu olmazdı abort çağırılır.

- terminate default olarak abort'u çağırır.

- bizz set\_terminate'e bir fonksiyon adresi verdiğimizde abort yerine o func çağırılır.

- bir daha çağırıldığında, set\_terminate get ettiği adres, necfunc adresi → daha önce kaydedilmiş func addr.

```

int main()
{
    std::cout << "main çağrıldı\n";

    try {
        f1(); // burada throw!; ifadesi vardı
    }
    catch (int) {
        std::cout << "main fonksiyonu içinde hata yakalandı\n"; // Bu blok yönetildi
    }

    std::cout << "main sona erdi\n";
}

```

→ Tür dönüşümü ve integral promotion, catch parametrelerinde çalışmaz. Aynı tür olmalı.

→ Bir switch case gibi yapı var. İşlendikten sonra break. ve koda devam eder.

### \*Önemli İstisna:

- Catch parametremiz, base reference olduğunda, hem base, hem derived class ref. catch edilir.

```

catch(std::exception &ex)
{
    std::cout << ex.what() << "\n";
}

```

// Kopyalama işlemi yapmamak / object string'e yollanmamak için. class reference

std::exception // exception türlerinde virtual func.

// Farklı overloadları var.

### \*Exception Hiyerarşisi:

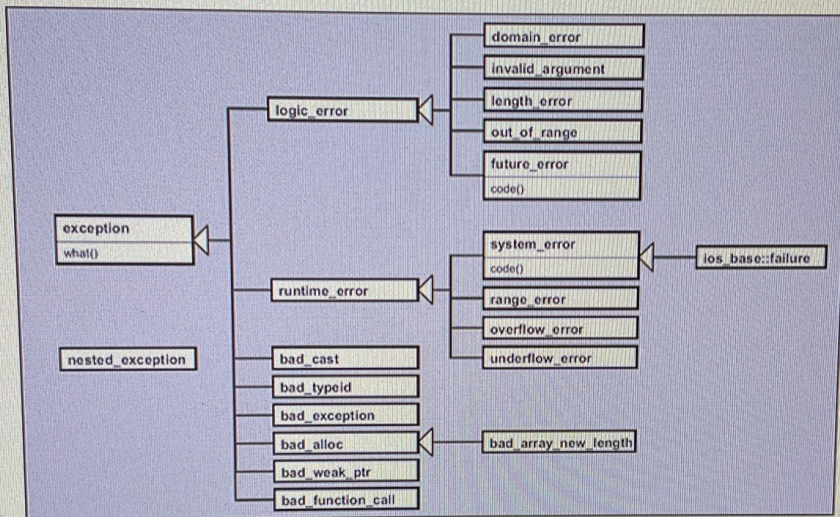


Figure 4.1. Hierarchy of Standard Exceptions

```

int main()
{
    try {
        func();
    }
    catch (const std::exception& ex) {
        //catch (const std::logic_error& ex) {
        //catch (const std::out_of_range& ex) {
            std::cout << "exception caught: " << ex.what() << '\n';
        }
    }
}

```

→ Bu da catch bloğu, aynı exception'ı yakalayabilir. Ancak

out of range → logic err → exception hiyerarşisi. Ver.



\* Fakat burada switen case gibi, en başta en sona teraerok girer. Bu yzden en bzeiden genelle dgru bir siraama yppilmeli

\* Heren her zaman throw ifadeleri, gzeri nesne oluatur gibi class egrmaa

```
void func()
{
    BadDate bd;

    throw bd;
}
```

Bu kod bd throw etmez X

→ Derleyicinin Bad Date sınıfı tarafından bir hata nesnesi olu-  
tması için Bad Date'in copy constructor'ına gazi yppilmeli.

```
void func()
{
    throw std::bad_alloc{};
}
```

\* Catch blounda ne yapacağını bilmiyarsan → yoleleme

\* Kaynagi catch bloau vermek → RAII idiyomundan ! datualar ile kaynaksizir. Bz , bu datualar i cagurmayi garantiyealır.

\* Exception Translate: - Exception ykalanır, bozu reeler yapılır ya da yppilmez, fakat aynı class tarafından dgrı, kendr seclayma  
kendir cektirmen. onayagı dıktır basta exception throw edilmek

\* Exception Rethrow: - Yakalanan exception, kendr tarafından exception gonderir.

```
void func()
{
    try {

    }
    catch (const std::exception& ex) {
        //...
        throw;
    }
}
```

→ hta bir class belirtmezse,  
rethrow edilir. Syntax bu sekilde

Fakat karıstırma

```
void func()
{
    try {

    }
    catch (const std::exception& ex) {
        //...
        throw ex;
    }
}
```

Bu rethrow dgrı !!