**2023.10.13**

**class** Nec **{**

**unsigned** **char\*** buf**[**1024**\***1024**]{};**

**};**

**int** main**()**

**{**

std**::**vector**<**Nec**\*>** nvec**;**

**int** i**{};**

**try**

**{**

**for** **(**i **=** 0**;** **,** i **<** 1'000'000**;** **++**i**)**

**{**

nvec**.**push\_back**(new** Nec**());**

**}**

**}**

***//***

**catch** **(const** std**::**bad\_alloc**&** ex**)**

**{**

std**::**cout **<<** "exception caugth" **<<**ex**.**what**()** **<<** "\n"**;**

std**::**cout **<<** "i = " **<<** i **<<** "\n"**;**

**}**

**catch** **(const** exception**&** ex**)**

**{**

std**::**cout **<<** "exception caugth" **<<**ex**.**what**()** **<<** "\n"**;**

std**::**cout **<<** "i = " **<<** i **<<** "\n"**;**

**}**

**}**

**int** main**()**

**{**

**using** **namespace** std**;**

string str**(**10**,** 'A'**);**

**try**

**{**

**auto** c **=** str**[**463**];** ***// undefined behavior***

**auto** c **=** str**.**at**(**463**);** ***// exception -> out\_of\_range***

**}**

***// catch blokları özelden genele doğru sıralanmalı***

**catch** **(const** std**::**out\_of\_range**&** ex**)**

**{**

std**::**cout **<<** "out of range caugth" **<<**ex**.**what**()** **<<** "\n"**;**

**}**

**catch** **(const** std**::**exception**&** ex**)**

**{**

std**::**cout **<<** "exception caugth" **<<**ex**.**what**()** **<<** "\n"**;**

**}**

**}**

exception yakalayınca hangi senoryalar devreye girer:

1. resumption (kaldığı yerden devam etme)
   * 1. resource leak olmıcak (database bağlantısı kopacak vb)
2. terminate (sonlandırma)
3. kısmi müdahale gerçekleştirip rethrow ediyoruz
4. exception translate etmek

**int** foo**();**

**int** main**()**

**{**

**using** **namespace** std**;**

**try**

**{**

**int** x **=** foo**();**

**}**

**catch** **(const** std**::**exception**&** ex**)**

**{**

***// hatalı olur x kullanamayız çünkü try scope'ta***

std**::**cout **<<** "hata yakalandi x = " **<<** x **<<** "\n"**;**

**}**

**}**

/\*

a) gönderilen exception'lar yakalanıyor mu

b) hataların yakalanmasi durumunda kaynak sızıntısı oluyor mu?

\*/

**class** Myclas

**{**

***//...***

**};**

**void** foo**();**

**void** bar**();**

**void** func**()**

**{**

Myclass **\***p **=** **new** Myclas**;**

***/\****

***exception verirse bu fonksiyonlar program akışı***

***func() fonksiyonundan çıkar ve p nesnesi delete***

***edilemez. Bu yüzden böyle kodlardan uzak durmalıyz***

***\*/***

foo**();**

bar**();**

**delete** p**;**

**}**

***/\****

***RAII***

***stack unwinding ( yığının geri sarımı)***

***f1 --> f2 --> f3 --> f4 --> f5***

***\*/***

**class** Myclas

**{**

**public:**

**~**Myclass**()**

**{**

std**::**cout **<<** **this** **<<** "kaynaklar geri verildi\n"**;**

**}**

**private:**

**int** ar**[**100**]{};**

**}**

**void** f4**()**

**{**

Myclass m4**;**

**throw** std**::**exception**{};**

**}**

**void** f3**()**

**{**

Myclass m3**;**

f4**();**

**}**

**void** f2**()**

**{**

Myclass m2**;**

f3**();**

**}**

**void** f1**()**

**{**

Myclass m1**;**

f2**();**

**}**

**int** main**()**

**{**

f1**();** ***// Myclass nesneleri böyle yaparsak destroy edilmesi***

**try**

**{**

***// Otomatik ömürlü oldukları için***

f1**();** ***// Myclass nesneleri destroy edilir.***

**}**

**catch** **(const** std**::**exception**&** ex**)**

**{**

std**::**cout **<<** "exception caugth" **<<** ex**.**what**()** **<<** "\n"**;**

**}**

**}**

**class** File

**{**

**public:**

File**(const** **char\*);**

**~**File**();**

**};**

**void** foo**();**

**void** bar**();**

**void** func**()**

**{**

File myfile**(**"necati.txt"**);**

***/\****

***exception verse bile aşağıdaki fonksiyonlar***

***myfile otomatik ömürlü olduğu için yine stack***

***unwinding aşamasında destroy edilir.***

***\*/***

foo**();**

bar**();**

**}**

Aksi biçimde davranmanıza gerekecek olağan dışı bir durum yokise dinamik ömürlü nesneleri ya da genel olarak kaynakları ham pointer'lar ile değil smart pointer nesneleri ile kullanın.

**Exception Rethrow Edilmesi**

**try**

**{**

foo**();**

**}**

***// eğer ex adlı parametreyi kullanmıcaksak isim verme***

**catch(const** std**::**MathError**&** ex**)**

**{**

***/\* eğer böyle yaparsak***

***1) copy ctor ile yeni ex oluşturulur***

***sonra eski ex destroy edilir.***

***2) ex diveded by zero olsa bile throw ex***

***yaparsak MathError exception döner yani***

***object slicing olur***

***\*/***

**throw** ex**;**

***// rethrow böyle yapılır***

**throw;**

**}**

***// rethrow***

**try**

**{**

foo**();**

**}**

**catch(const** std**::**MathError**&** ex**)**

**{**

**throw;** ***// yakalanan nesne ile gönderilen nesne aynı***

**}**

**void** func**()**

**{**

**try**

**{**

**throw** std**::**out\_of\_range **{**"range hatasi"**};**

**}**

**catch** **(const** std**::**exception**&** ex**)**

**{**

std**::**cout **<<** "func içinde hata." **<<** ex**.**what**()** **<<** "\n"**;**

**throw** ex**;** ***// std::exception bloğuna girer (object slicing)***

**throw;** ***// out\_of\_range bloğuna girer.***

**}**

**}**

**int** main**()**

**{**

**try**

**{**

func**();**

**}**

**catch** **(const** std**::**out\_of\_range**&)**

**{**

std**::**cout **<<** "out\_of\_range"**;**

**}**

**catch** **(const** std**::**exception**&)**

**{**

std**::**cout **<<** "hata yakalandi"**;**

**}**

**}**

**void** foo**()**

**{**

**if** **(**1**)**

**{**

***// hatalı çünkü default ctor yok***

**throw** std**::**out\_of\_range**{};**

**}**

**}**

**void** bar**()**

**{**

**throw;**

**}**

**void** foo**()**

**{**

**try**

**{**

**if** **(**1**)**

**throw** std**::**out\_of\_range**(**"out of range error"**);**

**}**

**catch** **(const** std**::**exception**&** ex**)**

**{**

std**::**cout **<<** "exception caugth: " **<<** ex**.**what**()** **<<** "\n"**;**

bar**();** ***// rethrow eder***

**}**

**}**

**void** bar**()**

**{**

**throw;**

**}**

**int** main**()**

**{**

bar**();** ***// terminate fonksiyonu çağrılır.***

**}**

***/\****

***Eğer bir rethrow statemant yürütüldüğünde yakalanmış***

***bir hata nesnesi yok ise std::terminate çağrılır.***

***\*/***

**void** bar**()**

**{**

**throw;**

**}**

**int** main**()**

**{**

**try**

**{**

bar**();** ***// terminate edilir catch'e girmez***

**}**

**catch** **(...)**

**{**

std**::**cout **<<** "exception caugth: "**;**

**}**

**}**

**Exception in CTOR**

**class** Myclass

**{**

**public:**

Myclass**(int** x**)** **:** np**(new** **int[**100**])**

**{**

**if** **(** x **<** 0**)**

**throw** std**::**invalid\_argument**{**"gecersiz arguman"**};**

**}**

**~**Myclass**()**

**{**

std**::**cout **<<** "Myclas desturctor\n"**;**

**delete[}** np**;**

**}**

**private:**

**int** **\***mp**;**

**}**

**int** main**()**

**{**

**try**

**{**

Myclas m**{-**12**};** ***// dtor çağrılmaz***

***/\****

***dtor çağrılmaz çünkü bir sınıf nesnesinin***

***hayata gelmesi için ctor kodunun tamamı tamamlanması gerekiyor***

***\*/***

**}**

**catch** **(const** std**::**exception**&** ex**)**

**{**

std**::**cout **<<** "exception caugth: " **<<** ex**.**what**()** **<<** "\n"**;**

**}**

**}**

**class** A

**{**

**public:**

A**(int)**

**{**

std**::**cout **<<** "A(int) ctor\n"**;**

**}**

**~**A**()**

**{**

std**::**cout **<<** "A dtor\n"**;**

**}**

**};**

**class** B

**{**

**public:**

B**(int)**

**{**

std**::**cout **<<** "B(int) ctor\n"**;**

**}**

**~**B**()**

**{**

std**::**cout **<<** "B dtor\n"**;**

**}**

**};**

**class** Myclass

**{**

**public:**

Myclas**()** **:** ax**(**0**),** bx**(**1**)**

**{**

**if(**1**)**

**{**

**throw** std**::**bad\_alloc**{};**

**}**

**}**

**~**Myclass**()**

**{**

std**::**cout "Myclass dtor\n"**;**

**}**

**private:**

A ax**;**

B bx**;**

**}**

**void** foo**()**

**{**

Myclas m**;**

**}**

**int** main**()**

**{**

**try**

**{**

foo**();**

***/\****

***ax ve bx nesnelerinin dtor çağrılır çünkü***

***hayata gelmiş durumundalar ancak myclass nesnesi***

***hayata gelmediği için onun dtor çağrılmaz***

***\*/***

**}**

**catch** **(const** std**::**exception**&** ex**)**

**{**

std**::**cout **<<** "exception: " **<<** ex**.**what**()** **<<** "\n"**;**

**}**

**}**

**class** Member

**{**

Member**()**

**{**

std**::**cout **<<** "Member ctor\n"**;**

**}**

**~**Member**()**

**{**

std**::**cout **<<** "Member ctor\n"**;**

**}**

**};**

**class** Nec

**{**

**public:**

Nec**()** **:** mp**{new** Member**}**

**{**

**throw** 1**;**

**}**

**~**Nec**()** **{**

**delete** mp**;**

**}**

**private:**

Member**\*** mp**;**

***//std::unqiue\_ptr<Member> mp;***

**};**

**int** main**()**

**{**

**try**

**{**

Nec mynec**;**

**}**

**catch(int** **)**

**{**

***// Member dtor cagrılmaz cünkü heap'te oluşuyor***

std**::**cout **<<** "exception caught\n"**;**

**}**

**}**

**class** Member

**{**

**public:**

Member**(int** x**)**

**{**

std**::**cout **<<** "Member ctor\n"**;**

**if** **(**x **<** 0**)**

**{**

**throw** std**::**runtime\_error**{**"exception from Member"**};**

**}**

**}**

**~**Member**()**

**{**

std**::**cout **<<** "Member ctor\n"**;**

**}**

**};**

**class** Nec

**{**

Nec**(int** val**)** **:** mx**(**val**)**

**{**

***// bu exception yakalanamaz***

**try**

**{**

**}**

**catch** **(...)**

**{**

std**::**cout **<<** "hata yakalandi\n"**;**

**}**

**}**

**private:**

Member mx**;**

**};**

**int** main**()**

**{**

Nec mynec**(-**23**);**

**}**

**Function Try Block**

**void** func**(int** x**)**

**{**

**try**

**{**

***// all function code***

**}**

**catch** **(const** std**::**exception**&)**

**{**

**}**

**}**

***// yukarıdaki yerine***

**int** func**(int** x**)**

**try**

**{**

**int** y **=** 5**;**

***// all code***

**if** **(**x **<** 0**)**

**throw** std**::**runtime\_error**{** "gecersiz deger" **};**

**return** 3**;**

**}**

**catch** **(const** std**::**exception**&)**

**{**

*/// y = 5; hatalı*

**return** x **\*** 6**;**

**},**

**Function Try Blok in CTOR**

**class** Member

**{**

**public:**

Member**(int** x**)**

**{**

std**::**cout **<<** "Member ctor\n"**;**

**if** **(**x **<** 0**)**

**{**

**throw** std**::**runtime\_error**{** "exception from Member" **};**

**}**

**}**

**~**Member**()**

**{**

std**::**cout **<<** "Member ctor\n"**;**

**}**

**};**

**class** Nec

**{**

Nec**(int** val**)** **try** **:** mx**(**val**)**

**{**

***// function try block eski cpp'de var***

**}**

**catch** **(const** std**::**exception**&** ex**)**

**{**

***// bu sefer Nec sınıfındaki exception yakalanır***

std**::**cout **<<** "exception caught: " **<<** ex**.**what**()** **<<** "\n"**;**

***// yine de kod patlar çünkü derleyici patlatır Nec nesnesi oluşamadığı için***

**}**

**private:**

Member mx**;**

**};**

**int** main**()**

**{**

Nec mynec**(-**23**);**

**}**

**class** A

**{**

**public:**

A**()** **=** **default;**

A**(const** A**&)**

**{**

std**::**cout **<<** "A copy ctor\n"**;**

**throw::**std**::**runtime\_error**{**"error from copy ctr of class A\n"**};**

**}**

**};**

**class** Nec

**{**

**public:**

Nec**(**A**)** **try**

**{**

**}**

**catch** **(const** std**::**exception**&** ex**)** **{**

std**::**cout **<<** "exception caught..." **<<** ex**.**what**()** **<<** "\n"**;**

**}**

**};**

**int** main**()**

**{**

***// copy ctor'dan gelen exception yakalayamaz***

A ax**;**

Nec mynec**(**ax**);**

**}**

**Exception Gurantee(s)**

**Basic gurantee**:exception gönderildiğinde bir kaynak sızıntısı olmıcak bu garanti altına alınacak ve nesne veya nesnelerin state'i geçerli olacak.

**Strong gurantee**: Basic gurantee özelliklerini veriyor ve state sadece geçerli değil ayrıca state aynı değişmiyor (commit or rollback) ya işini yap ya da işe başlamadan önceki state'i koru

***try***

***{***

***std::vector<int> x {1,2,3,4};***

***}***

***catch***

***{***

***burda exception yakalandı ve x'in size değişti 10 dan 12'ye çıktı***

***basic gurantee ama size değişmedi strong gurantee***

***}***

**Nothrow gurantee:** exception kesinlikle gönderilmicek derleyici buna göre yaptığı optimazsayonu değiştiriyor