**2023.10.11**

**String View**

/\*

class StringView

{

public:

// string sınıfı fonksiyonları

private:

// kopyalama yapmadan bu iki pointer ile o str'yi kontrol edebilirim

const char ps\*;

const char pe\*

}

\*/

#include <string>

#include <string\_view>

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

**{**

std**::**string\_view sv **{** "necati ergin" **};**

***// okuma fonksiyonlarına erişebiliriz ve gözlemci olabiliyoruz***

**}**

**String Sınıfı Toplama Operatorü**

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

**{**

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

string name **{** "mert" **};**

string surname **{** "kaptan" **};**

**auto** s **=** name **+** ' ' **+** surname**;**

***// operator+(operator+(name, ' '),surname);***

**}**

**String Tür Dönüşümleri**

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

**{**

**int** ival**{};**

**double** dval**{};**

**auto** istr **=** to\_string**(**ival**);**

**auto** dstr **=** to\_string**(**dval**);**

string str**;**

**int** x **=** 56**;**

str **=** x**;** ***// operator=(char) tür dönüşümü olmaz***

***//string to int, long, double vb***

***/\****

***stoi***

***stoul***

***stol***

***stod***

***\*/***

string s **{**"23243"**};**

**auto** ival **=** stoi**(**s**);** ***// int ival = 23243***

string s1 **{**"3213emre"**};**

**size\_t** idx**{};**

**auto** ival1 **=** stoi**(**s1**,** **&**idx**);** ***// int ival1 = 32133***

***// idx kullanılmayan verinin size döndürür***

string s2 **{**"eee3213emre"**};**

**auto** ival2 **=** stoi**(**s1**);** ***// exception verir***

**}**

**Exception Handling**

***/\****

***assertions (doğrulama) (C)***

***dynamic assertions (runtime)***

***static assertions (compiler time)***

***assert( x > 0)***

***void func(int \*ptr)***

***{***

***assert(ptr != NULL);***

***assert(b !=0);***

***auto x = a / b;***

***}***

***exception : (run-time error)***

***try***

***{***

***f1(); // f1 ve içindeki her fonksiyon bu bloğa dahil***

***}***

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

***{***

***}***

***catch(long)***

***{***

***}***

***bir hata nesnesi gönderildi ve bu hata nesnesi yakalamadi bu durumda ne olur?***

***Bu duruma uncaught expcetion denir***

***std::terminate***

***abort***

***terminate fonksiyonu uncaught exception durumunda abort fonksiyonu çağırır***

***ancak bizim istediğim fonksinu çağırması için set\_terminate fonks kullanılır.***

***void terminate(void) {}***

***set\_terminate(void (\*)(void))***

***\*/***

**class** ExBase

**{**

**public:**

**virtual** **const** **char\*** what**()const**

**{**

**return** "exbase"**;**

**}**

**}**

**class** MathException **:** **public** Exbase

**{**

**public:**

**const** **char\*** what**()const** **override**

**{**

**return** "exbase"**;**

**}**

**}**

**class** DivideByZero **:** **public** MathException

**{**

**public:**

**const** **char\*** what**()const** **override**

**{**

**return** "exbase"**;**

**}**

**}**

**void** myabort**()**

**{**

std**::**cout **<<** "myabort cagrildi\n"**;**

exit**(**EXIT\_FAILURE**);**

**}**

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

**{**

std**::**cout **<<** "f4 cagrildi\n"**;**

***//throw 1 // exception atıyoruz***

**throw** DivideByZero

std**::**cout **<<** "f4 sona erdi\n"**;**

**}**

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

**{**

std**::**cout **<<** "f3 cagrildi\n"**;**

f4**();**

std**::**cout **<<** "f3 sona erdi\n"**;**

**}**

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

**{**

std**::**cout **<<** "f2 cagrildi\n"**;**

f3**();**

std**::**cout **<<** "f2 sona erdi\n"**;**

**}**

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

**{**

std**::**cout **<<** "f1 cagrildi\n"**;**

f2**();**

std**::**cout **<<** "f1 sona erdi\n"**;**

**}**

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

**{**

set\_terminate**(&**myabort**);**

std**::**cout **<<** "main basladi\n"**;**

**try**

**{**

f1**();**

**}**

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

**{**

std**::**cout **<<** "hata yakalandi catch(int)"**;**

**}**

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

**{**

std**::**cout **<<** "hata yakalandi catch(unsigned int)"**;**

**}**

**catch** **(double)**

**{**

std**::**cout **<<** "hata yakalandi catch(double)"**;**

**}**

**catch** **(const** Exbase**&** ex**)** ***// her zaman refarans olacak***

**{**

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

**}**

std**::**cout **<<** "main sonaerdi\n"**;**

**}**

catch her zaman referans olacak çünkü:

* copy ctor'dan korunmak için
* virtual dispacth faydalanmak için

***/\****

***throw expr;***

***int x = 5;***

***throw x;***

***derleyici şöyle bir kod oluşturur.***

***int exception\_object{x};***

***ya da***

***throw 12;***

***int exception\_object{12};***

***\*/***

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

**{**

std**::**cout **<<** "f4 cagrildi\n"**;**

std**::**string s **{**"emre bahtiyar"**};**

***// out\_of\_range --> logic\_error --> exception***

**auto** c **=** str**.**at**(**565**);** ***// exception***

std**::**cout **<<** "f4 sona erdi\n"**;**

**}**

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

**{**

std**::**cout **<<** "f3 cagrildi\n"**;**

f4**();**

std**::**cout **<<** "f3 sona erdi\n"**;**

**}**

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

**{**

std**::**cout **<<** "f2 cagrildi\n"**;**

f3**();**

std**::**cout **<<** "f2 sona erdi\n"**;**

**}**

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

**{**

std**::**cout **<<** "f1 cagrildi\n"**;**

f2**();**

std**::**cout **<<** "f1 sona erdi\n"**;**

**}**

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

**{**

set\_terminate**(&**myabort**);**

std**::**cout **<<** "main basladi\n"**;**

**try**

**{**

f1**();**

**}**

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

**{**

***//exception ile std'in tüm hataları yaklayabiliriz***

***// logic\_error ile logic hataları yakalarız.***

***// out\_of\_range ile sadece bunla ilgili hataları***

**}**

**catch(...)**

**{**

***// bütün hataları yakalar***

**}**

**}**