**2023.09.29**

**initializer\_list**

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

**{**

std**::** initializer\_list**<int>** x**{** 1**,** 2**,** 3**,** 4**,** 5**};**

***/\****

***const int ar[] = { 1, 2, 3, 4, 5};***

***class initializer\_list***

***{***

***const int \*ps; başlangınç adresi***

***const int \*pe; son adresi***

***}***

***\*/***

**auto** x **=** **{** 1**,** 2**,** 3**,** 4**,** 5**};** ***// x'in türü initializer\_list<int>***

**auto** x **{** 1**,** 2**,** 3**,** 4**,** 5**};** ***// syntax hatası***

**auto** x **=** **{**1**};** ***// x'in türü initializer\_list<int>***

**auto** x**{**1**};** ***// x'in türü int***

**}**

**class** Myclass

**{**

**public:**

Myclass**(**std**::**initializer\_list**<int>)**

**{**

std**::**cout **<<** "Myclass(initializer\_list)\n"**;**

**}**

Myclass**(int)**

**{**

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

**}**

Myclass**(int,** **int)**

**{**

std**::**cout **<<** "Myclass(int, int)\n"**;**

**}**

**};**

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

**{**

Myclass m1**(**10**,** 20**);** ***// return Myclass(int, int)***

Myclass m2**{**10**,** 20**};** ***// return Myclass(initializer\_list)***

**}**

***//vector<>(size\_t)***

***//vector<>(initializer\_list)***

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

**{**

std**::**vector**<int>** vec1**(**100**);** ***// size\_t - 100 tane 0 sıfır değeri ile başlatcak***

std**::**vector**<int>** vec2**{**100**};** ***// initializer\_list - 1 tane 100 değeri ile başlatcak***

**}**

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

**{**

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

string s1**(**50**,** 'A'**);** ***// 50 tane A karakteriyle başlatcak***

string s1**{**50**,** 'A'**};** ***// 50'in ascii değeri ve A ile başlatcak***

**}**

**String Class**

***/\****

***std::string::size\_type //size\_t türüne eş***

***std::string::npos // türü size\_type***

***size\_type:***

***yazı uzunluğu türü***

***kapasite türü***

***bazı string fonksiyonların istediği indeks değeri***

***string sınıfındaki fonksiyonların aldığı arguments:***

***const string&***

***const string&, size\_type idx : idx indeksinden sonraki string boyunca işlem yapacak***

***const string&, size\_type idx, size\_type len : idx'ten başlayarak len kadar işlem yapacak***

***const char\* (cstring) : null karaterden biz sorumluyuz***

***const char\*, size\_type len***

***char***

***size\_type, char c : n tane c karakteri (fill)***

***iterator beg, iterator end : ramge parametre***

***std::initializer\_list<char>***

***\*/***

**String Sınıfı Ctor'lar ve Fonksiyonları**

***//string sınıfı ctor'lar ve fonksiyonları***

**void** ps**(const** std**::**string**&** s**)**

**{**

std**::**cout **<<** "|" **<<** s **<<** "|\n"**;**

**}**

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

**{**

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

string s1**;** ***//Default ctor***

ps**(**s1**);** ***// boş yazı çıkar***

***// s1.size() ve s1.length() return type: size\_type***

cout **<<** "s1.size() = " **<<** s1**.**size**()** **<<** "\n"**;** ***// common container interface***

cout **<<** "s1.length() = " **<<** s1**.**length**()** **<<** "\n"**;**

s1**.**empty**()** ***// is empty()***

***// cstring ctor***

string str**(**"emre bahtiyar"**);** ***// cstring ctor***

**char** ar**[]** **=** **{** 'A'**,** 'B'**,** 'C'**};**

string str**(**ar**);** ***// tanımsız davranış***

ar**[]** **=** "murathan"**;**

string str**(**ar **+** 5 **);** ***// return han --cstring ctor***

***//const char\*, size\_type len ctor***

string str**(**ar **,**3**);** ***// return mur (const char\*, size\_type len )***

string str**(**ar**+**3 **,** 2**);** ***// return at***

string str**(**ar**,** 20**);** ***// tanımsız davranış***

string str**(**ar**,** 6**);** ***// return murat + null karakter***

std**::**string str**(**'A'**)** ***// parametresi char olan ctor yok hatalı***

***//substring ctor***

string s1 **{**"cengizhan"**};**

string s2 **(**s1**,** 3**);** ***//substring ctor : return gizhan***

string s2 **(**s1**,** 3**,** 50**);** ***// geriye kalanların hepsini alınır - hata yok***

**}**

***/\****

***String CTOR:***

***string s1("a") cstring ctor***

***string s2(13, 'a') fill ctor***

***string s3{'A'} initializer\_list ctor***

***string s4(s2, 3) substring ctor***

***\*/***

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

**{**

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

ifstream ifs **{**"main.c"**};**

vector**<**string**>** svec**;**

string sline**;**

**while** **(**getline**(**ifs**,** sline**))**

**{**

***//svec.push\_back(sline);***

svec**.**push\_back**(**std**::**move**(**sline**));** ***// sline geri kullanılabilir***

**}**

**}**

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

**{**

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

**char** str**[]** **=** **{** "gokhan girgin" **};**

string s1 **{**str**};** ***//cstr ctor***

string s2 **{** str **,** 3**};** ***// array ctor***

string s3 **{** str **,** str **+** 4**};** ***// range ctor***

**}**

**capacity()const : return size\_type**

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

**{**

**using** namaspace std**;**

string str**(**153**,** 'A'**);**

cout **<<** "str.size() = " **<<** str**.**size**()** **<<** "\n"**;** ***// return 153***

cout **<<** "str.capacity() = " **<<** str**.**capacity**()** **<<** "\n"**;** ***// return 159***

***/\****

***size 153 eğer 6 tane daha karakter eklersem capacity dolacak ve***

***reallaction olacak.***

***String ve vector sınıflarında kapasitenin artış katsayısı derleyiciye***

***bağlıdır.***

***\*/***

**}**

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

**{**

***/\****

***kötü kod çünkü sürekli reallaction yapmış oluyuruz. Baştan size***

***belirtseydik reallaction olmıcaktı.***

***str.reserve(1200); // böyle kapasiteyi önceden belirlemiş olduk***

***\*/***

string str**;**

**char** c **=** 'S'**;**

**for** **(int** i **=** 0**;** i **<** 1000**;** **++**i**)**

**{**

str**.**push\_back**(**c**)**

**}**

**}**

**Small String Optimazsayonu**

Derleyeciler belli bir boyutta kadar allaction yapmaz ve kendi içide oluşturur