**2023 11 08**

**Vector Silme Fonksiyonları**

***/\****

***erase (iterator,range)***

***pop\_back***

***clear***

***\*/***

**pop\_back**

***// pop\_back***

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

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

**{**

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

rfill**(**svec**,** 10**,** rname**);**

**while** **(!**svec**.**empty**())**

**{**

print**(**svec**);**

svec**.**pop\_back**();** ***// sondan eleman siler***

svec**.**erase**(**svec**.**begin**());** ***// iterator olur ( silinmiş öğeden sonraki ilk öğeyi döndürür)***

**}**

**}**

**Aranan İlk Öğeyi Silmek**

***// aranan ilk öğeyi silmek***

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

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

**{**

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

rfill**(**svec**,** 10**,** rname**);**

string name**;**

cout **<<** "silenecek öğe:"**;**

cin**>>** name**;**

**if** **(const** **auto** iter **=** find**(**svec**.**begin**(),** svec**.**end**(),** name**);** iter **!=** svec**.**end**())**

**{**

svec**.**erase**(**iter**);**

cout **<<** "bulundu ve silindi\n"**;**

**}**

**else**

**{**

cout **<<** "bulunamadı\n"**;**

**}**

**}**

***// içinde hangi harf bulunan silinsin***

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

**{**

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

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

rfill**(**svec**,** 10**,** rname**);**

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

**if** **(const** **auto** **=** iter find\_if**(**svec**.**begin**(),** svec**.**end**(),**

**[**c**](const** string**&** s**)** **{return** s**.**contains**(**c**);};** iter **!=** svec**.**end**())**

**{**

svec**.**erase**(**iter**);**

**}**

**}**

***// sondan ilk elemanı silme***

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

**{**

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

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

rfill**(**svec**,** 10**,** Irand**(**0**,** 4**);**

**int** ival**;**

cout **<<** "silenecek öğe:"**;**

cin**>>** ival**;**

**if** **(const** **auto** iter **=** find**(**ivec**.**rbegin**(),** ivec**.**rend**(),** ival**);**

iter **!=** ivec**.**rend**())**

**{**

***// iter.base() -> iter.end() sarmalar***

ivec**.**erase**(**iter**.**base**()** **-** 1**);**

**}**

**}**

***//erase range***

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

**{**

vector**<**string**>** svec**{**"nuri"**,** "emre"**,** "gano"**,** "ahmet"**};**

svec**.**erase**(**iter**.**begin**(),** iter**.**end**());** ***// hepsi silinir***

svec**.**erase**(**iter**.**begin**(),** next**(**iter**.**begin**(),** 3**));** ***// ilk 3 silinir***

svec**.**erase**(**next**(**iter**.**begin**(),** iter**.**end**()** **-** 1**));** ***// başta ve sondakiler kalır***

**}**

***// clear***

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

**{**

vector**<**string**>** svec**{**"nuri"**,** "emre"**,** "gano"**,** "ahmet"**};**

svec**.**empty**();**

svec**{};**

svec**.**resize**(**0**);**

**}**

**shrink to fit**

#include <format>

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

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

**{**

vector**<int>** ivec**{**100'000**,** 7**};**

std**::**cout **<<** format**(**"size = {1}, capacity = {0}\n"**,**

ive**.**capacity**(),** ivec**.**size**());**

ivec**.**erase**(**ivec**.**begin**()** **+** 5 **,** ivec**.**end**());**

ivec**.**shrink\_to\_fit**();** ***// capacity geri verir***

**}**

**vector.data()**

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

**void** array\_print**(const** **int\*** p**,** **size\_t** size**)**

**{**

**while(**size**--)**

print**(**"%d"**,**p**++);**

**}**

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

**{**

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

vector**<int>** ivec**{** 3**,** 6**,** 7**,** 9**,** 1**,** 10**};**

array\_print**(**ivec**.**data**(),** ivec**.**size**());**

***// vector'deki ilk öğenin adresi***

**int** **\***p1 **=** ivec**.**data**();**

**int** **\***p2 **=** **&**ivec**[**0**];**

**int** **\***p3 **=** **&\***ivec**.**begin**();**

vector**<int>** ivec1**;** ***// boş container***

**auto** p **=** ivec**.**data**();**

cout **<<** **(**p **==** **nullptr);** ***// true***

**}**

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

**{**

vector**<**string**>** s1**(**100'000**);**

vector**<**string**>** s2**(**100'000**);**

***// pointerler swap ediliyor***

s1**.**swap**(**s2**);**

swap**(**s1**,** s2**);**

***// böyle yapmamalıyız***

**auto** temp **=** s1**;**

s1 **=** s2**;**

s2 **=** temp**;**

**}**

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

**{**

vector**<bool>** myvec**(**20**);**

**auto** b **=** myvec**[**0**];** ***// b'nin türü std::vector<bool>::reference***

**}**

**template<typename** T **...>**

vector**<bool** A**<>>**

**{**

**class** reference

**{**

**operator** **bool()const** **noexcept;**

**void** flip**();**

**operator=(bool);**

**};**

reference **operator[](size\_t** idx**);**

reference at**(size\_t** idx**);**

reference front**();**

reference back**();**

**}**

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

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

**{**

vector**<bool>** myvec**(**20**);**

vector**<bool>::**reference x **=** myvec**.operator[](**5**);** ***//myvec[5];***

***// Dikkat !***

**bool** b **=** myvec**[**4**];** ***// bool olur***

**auto** x **=** myvec**[**3**];** ***// std::vector<bool>::reference olur***

**}**

**STL'deki silme algoritmaları**

***/\****

***remove***

***remove\_if***

***unique***

***\*/***

**std::remove**

**template<typename** Iter**,** **typename** T**>**

Iter Remove**(**Iter beg**,** Iter end**,** **const** T**&** val**)**

**{**

**}**

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

**{**

vector**<**string**>** svec **{** "ali"**,** "can"**,** "eda"**,** "can"**,**

"emre"**,** "gano"**,** "can"**};**

cout **<<** "size = " svec**.**size**()** **<<** "\n"**;** ***// size = 7***

**auto** logic\_end\_iter **=** remove**(**svec**.**begin**(),** svec**.**end**(),** "can"**);**

cout **<<** "size = " svec**.**size**()** **<<** "\n"**;** ***// size = 7***

std**::**cout **<<** "silinmemiş öge sayısı" **<<**

distance**(**svec**.**begin**(),** logic\_end\_iter**);** ***// 4***

std**::**cout **<<** "silinmiş ögesi sayısı" **<<**

distance**(**logic\_end\_iter**(),** svec**.**end**());** ***// 3***

***// logic olarak silinmiş öğeleri silme işlemi***

svec**.**erase**(**logic\_end\_iter**,** svec**.**end**());**

cout **<<** "size = " **<<** svec**.**size**());** ***// size = 4***

**}**

**Erase Remove İdiom**

***// erase remove idiom***

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

**{**

vector**<**string**>** svec **{** "ali"**,** "can"**,** "eda"**,** "can"**,**

"emre"**,** "gano"**,** "can"**};**

svec**.**erase**(**remove**(**svec**.**begin**(),** svec**.**end**(),** "can"**),** svec**.**end**());**

cout **<<** "size = " **<<** svec**.**size**());** ***// size = 4***

***// cpp20 ile bu idiom yerine erase fonksiyonu var***

**auto** n **=** erase**(**svec**,** "can"**);** ***// n: silinen öğe sayısı***

**}**

**std::remove\_if and std::erase\_if**

***// erase remove idiom***

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

**{**

vector**<**string**>** svec **{** "ali"**,** "can"**,** "eda"**,** "can"**,**

"emre"**,** "gano"**,** "can"**};**

svec**.**erase**(**remove**(**svec**.**begin**(),** svec**.**end**(),** "can"**),** svec**.**end**());**

cout **<<** "size = " **<<** svec**.**size**());** ***// size = 4***

***// cpp20 ile bu idiom yerine erase fonksiyonu var***

**auto** n **=** erase**(**svec**,** "can"**);** ***// n: silinen öğe sayısı***

**}**

***// remove\_if and erase\_if***

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

**{**

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

rfill**(**svec**,** 2000**,** rname**);**

**size\_t** len **=** 10**;** ***// 10'dan büyük olanlar***

svec**.**erase**(**remove\_if**(**svec**.**begin**(),** svec**.**end**(),[**len**](const** string**&** s**)**

**{return** s**.**size**()** **>** len**};),** svec**.**end**());**

***// cpp -20***

**auto** n **=** erase\_if**(**svec**,** **[**len**](const** string**&** s**)**

**{return** s**.**size**()** **>** len**});** ***// n: silinen öğe sayısı***

**}**

**template<typename** T**>**

**struct** Less

**{**

**bool** **operator()const(**T T**&** lhs**,** **const** T**&** rhs**)** **const**

**{**

**return** lhs **<** rhs**;**

**}**

**};**

**template<typename** T**>**

**struct** Plus

**{**

**bool** **operator()const(**T T**&** lhs**,** **const** T**&** rhs**)** **const**

**{**

**return** lhs **+** rhs**;**

**}**

**};**

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

**{**

Less**<int>{}(**12**,** 5**)** ***// 12 < 5 yazmaktan farkı yok***

**}**

**std::sort**

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

**{**

**using** name std**;**

vector**<int>** ivec**{** 455**,** 123 **,**1221**,** 545**,** 12**,** 1**,** 12**,** 43**,** 513**,** 123**,** 53**};**

sort**(**ivec**.**begin**(),** ivec**.**end**());**

sort**(**ivec**.**begin**(),** ivec**.**end**(),** less**<int>{});**

sort**(**ivec**.**begin**(),** ivec**.**end**(),** greater**<int>{});**

**}**

**std::transfrom**

#include <functional>

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

**{**

vector**<int>** x**{** 1**,** 32**,** 4**,** 1**,** 4**,** 5**,** 3**,** 1**,** 2**,** 5**,** 1**,** 7**};**

vector**<int>** y**{** 11**,** 32**,** 14**,** 12**,** 42**,** 53**,** 1**,** 2**,** 2**,** 5**,** 1**,** 7**};**

***// x ve y'yi toplayıp z'ye yazıyoruz***

transform**(**x**.**begin**(),** x**.**end**(),** y**.**begin**(),** back\_inserter**(**z**),** plus**{});**

vector**<int>** z**;**

**}**

#include <functional>

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

**{**

vector**<int>** x**{** 1**,** 32**,** 4**,** 1**,** 4**,** 5**,** 3**,** 1**,** 2**,** 5**,** 1**,** 7**};**

***// x'i negatif yapıyor***

transform**(**x**.**begin**(),** x**.**end**(),** x**.**begin**(),** negate**{});**

**}**

**std::unique**

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

**{**

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

vector**<int>** x**{** 1**,** 32**,** 4**,** 1**,** 4**,** 5**,** 3**,** 1**,** 2**,** 5**,** 1**,** 7**};**

***// ardaşık aynı olanları siler***

***// istediğimiz fonksiyonu kullanabilirz ardaşık öğlere bakarken***

x**.**erase**(**unique**(**x**.**begin**(),** x**.**end**(),** equal\_to**{}),** x**.**end**());**

**}**

**bir string'teki fazla boşlukları silme**

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

**{**

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

string str **=** " emre bahtiyar gano çalışkan "

str**.**erase**(**unique**(**str**.**begin**(),** str**.**end**(),** **[](char** c1**,** **char** c2**)**

**{**

**return** isspace**(**c1**)** **&&** isspace**(**c2**);**

**}),** str**.**end**());**

**}**

**std::ostream\_iterator**

***// ostream\_iterator***

***// copy ile standart outputta container yazma***

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

**{**

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

vector**<int>** ivec **{**1**,** 23**,** 2**,** 1**,** 4**,** 6**,** 1**};**

copy**(**ivec**.**begin**(),** ivec**.**end**(),** ostream\_iterator**<int>{**cout**,** "\n"**});**

**}**