**2023 11 10**

Insertion

vector: bir insert işlemi yaptığımız da o insert ettiğimiz konumdan önceki tutan pointerler etkilenmez ama reallacation olmama şartıyla yani kapasite artmadıysa

Erase

vector: silinecek öğeden herhangi birini gösteren noktalar invalid hale gelir ama öncekilere etkilenmez.

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

**{**

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

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

rfill**(**svec**,** 40**,** rtown**);**

***// uzunluğu 5 olanlar silinsin ve 6 olanlardan bir tane daha eklensin***

***// hatalı***

**for** **(auto** iter **=** svec**.**begin**();** iter **!=** svec**.**end**();** **++**iter**)**

**{**

***// iter invalid hale gelir***

**if** **(**iter**->**length**()** **==** 5**)**

**{**

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

**}**

**else** **if** **(**iter**->**length**()** **==** 6**)**

**{**

svec**.**insert**(**iter**,** **\***iter**);**

**}**

**}**

***// hata yok***

**auto** iter **=** svec**.**begin**();**

**while** **(**iter **!=** svec**.**end**())**

**{**

**if** **(**iter**->**length**()** **==** 5**)**

**{**

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

**}**

**else** **if** **(**iter**->**length**()** **==** 6**)**

**{**

iter **=** next**(**svec**.**insert**(**iter**,** **\***iter**),** 2**);**

**}**

**else**

**{**

**++**iter**;**

**}**

**}**

**}**

**std::deque**

* ekleme ve silme baştan ya da sondan olacaksa deque kullanabiliriz.
* iterator invalid hale gelmemesini istiyorsak kullanabiliriz.
* reallacation maliyetinden kaçınmak için kullanabiliriz.

**template** **<typename** T**,** **typename** A **=** std**::**allocator**<**T**>>**

**class** Deque

**{**

**};**

***///////***

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

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

**{**

deque**<int>** id**;**

Irand myrand**{** 0**,** 9999**};**

**for** **(int** i **=** 0**;** i **<** 100**;** **++**i**)**

**{**

**int** val **=** myrand**();**

**if** **(**val **%** 2 **==** 0**)**

id**.**push\_back**(**val**);**

**else**

id**.**push\_front**(**val**);**

**}**

**}**

Insertion

deque: ekleme işlemleri iki uçtan birinde yapılırsa pointer ya da referencelar etkilenmez ama iterator invalid hale gelir.

Erase

deque: silme işlemi iki uçtan birinde yapılırsa silinmiş öğenin iterator ve pointer invalid hale gelir. Diğerlerine bir şey olmaz

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

**{**

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

deque**<**string**>** sd**;**

rfill**(**sd**,** 10**,** **[]** **{return** rname**()** **+** " " **+** rfname**();});**

**auto&** elem **=** sd**[**7**];**

**auto** iter **=** next**(**sd**.**begin**()** **+** 7**);**

cout **<<** elem **<<** "\n"**;**

cout **<<** **\***iter **<<** "\n"**;**

sd**.**push\_back**(**"mustafa aksoy"**);**

cout **<<** elem **<<** "\n"**;**

cout **<<** **\***iter **<<** "\n"**;** ***// tanımsız davranıi***

**}**

**Sıralama Algoritmaları:**

***// sıralama algoritmaları***

***/\****

***sort***

***partial\_sort***

***stable sort***

***nth element***

***partition***

***stable partition***

***\*/***

**std::sort**

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

**{**

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

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

rfill**(**svec**,** 1'000'000**,** rname**);**

**auto** tp\_start **=** chrono**::**stead\_clock**::**now**();**

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

std**::**ofstream ofs**{**"out.txt"**};**

**if(!**ofs**)**

**{**

std**::**cerr **<<** "out.txt dosyasi olusturulmadi\n"**;**

**}**

copy**(**svec**.**begin**()** svec**.**end**(),** ostream\_iterator**<**string**>{**ofs**,** "\n"**});**

**auto** tp\_end **=** chrono**::**stead\_clock**::**now**();**

std**::**cout **<<** "siralama bitti: " **<<** chrono**::**duration**<double>(**tp\_end **-** tp\_start**)** **<<** "\n"**;**

**}**

**std::** **stable\_sort**

***//stable\_sort***

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

**using** sipair **=** std**::**pair**<**string**,** **int>**

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

**{**

vector**<**sipair**>** svec**;**

rfill**(**svec**,** 20'000**,** **[]{return** make\_pair**(**rname**()**i Irand**{** 20**,** 70**}(});));**

***// isme göre sıralama***

sort**(**svec**.**begin**(),** svec**.**end**(),** **[](const** sipair**&** x**,** **const** sipair**&** y**)**

**{**

**return** x**.**first **<<** y**.**first**;**

**});**

***// fonksiyon stable değil bu yüzden isimlerin sıralaması bozuldu***

sort**(**svec**.**begin**(),** svec**.**end**(),** **[](const** sipair**&** x**,** **const** sipair**&** y**)**

**{**

**return** x**.**second **<<** y**.**second**;**

**});**

***// isimlerde kendi içinde sıralıdır***

stable\_sort**(**svec**.**begin**(),** svec**.**end**(),** **[](const** sipair**&** x**,** **const** sipair**&** y**)**

**{**

**return** x**.**second **<<** y**.**second**;**

**});**

**}**

**std::partial\_sort and std::partial\_sort\_copy**

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

**{**

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

vector**<**Date**>** dvec**;**

rfill**(**dvec**,** 20'000**,** Date**::**random**);**

**int** n **=** 5'000**;**

***// ilk 5000 tanesini sıralar***

partial\_sort**(**dvec**.**begin**(),** dvec**.**begin**()** **+** n**,** dvec**.**end**());**

vector**<**Date**>** dest\_vec**(**n**);**

partial\_sort\_copy**(**dvec**.**begin**(),** dvec**.**end**(),**

dest\_vec**.**begin**(),** dest\_vec**.**end**());**

**}**

**nth element algoritması**

***// ortanca öğesini veya belirli bir yüzdelik dilimdeki öğeyi bulmak için kullanılır.***

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

**{**

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

vector**<**Date**>** dvec**;**

rfill**(**dvec**,** 20'000**,** Date**::**random**);**

**int** n **=** 1000**;**

***// nth\_element algoritması, bir dizideki sıralı olmayan öğeler***

***//arasında n'inci öğeyi bulmak için kullanılır.***

nth\_element**(**dvec**.**begin**(),** dvec**.**begin**()** **+** n**,** dvec**.**end**());**

**}**

**int** get\_median**(**std**::**vector**<int>** vec**)**

**{**

**auto** vec **{**r**};**

std**::**nth\_element**(**vec**.**begin**(),** next**(**vec**.**begin**(),** vec**.**size**()** **/** 2**),** vec**.**end**());**

**return** vec**[**vec**.**size**()** **/** 2**];**

**}**

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

**{**

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

vector ivec **{** 5**,** 7**,** 1**,** 4**,** 9**,** 3 **,** 23**,** 12**,** 8**,** 6**,** 13**,** **-**3**,** 98**};**

cout **<<** "median is " **<<** get\_median**(**ivec**);**

**}**

**std::partition**

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

**{**

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

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

rfill**(**svec**,** 20**,** rname**);**

**char** c **=** a**;**

***// içinde a harfi olanlarla olmayanları ayrır***

**auto** iter\_par\_point **=** partition**(**svec**.**begin**(),** svec**.**end**(),** **[**c**](const** string**&** s**)**

**{**

**return** s**.**find**(**c**)** **!=** string**::**npos**;**

**});**

print**(**svec**,** "\n"**);**

***// iter\_par\_point: koşulu sağlamayan ilk öğeyi verir***

std**::**cout **<<** "partisyon indeksi : " **<<**

iter\_par\_point **-** svec**.**begin**()** **<<** '\n'**;**

***// koşulu sağlayanlar***

copy**(**svec**.**begin**(),** iter\_par\_point**,** ostream\_iterator**<**string**>{**cout**,** "\n"**});**

***// koşulu sağlamayanlar***

copy**(**iter\_par\_point**,** svec**.**end**(),** ostream\_iterator**<**string**>{**cout**,** "\n"**});**

**}**