**2023 11 13**

**std::minmax\_element**

***// minmax\_element***

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

**{**

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

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

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

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

pair**<**vector**<**string**>::**iterator**,** vector**<**string**>::**iterator**>** ret

**=** minmax\_element**(**svec**.**begin**(),** svec**.**end**());** ***// std::pair döndür***

**auto** ip **=** minmax\_element**(**svec**.**begin**(),** svec**.**end**());**

cout **<<** "min = " **<<** **\***ip**.**first **<<** '\n'**;**

cout **<<** "max = " **<<** **\***ip**.**second **<<** '\n'**;**

***// Structured binding***

**auto** **[**iter\_min**,** iter\_max**]** **=** minmax\_element**(**svec**.**begin**(),** svec**.**end**());**

**}**

**std::partition\_point**

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

**{**

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

vector ivec**{** 2**,** 4**,** 6**,** 8**,** 2**,** 20**,** 4**,** 3**,** 7**,** 9**,** 101**};**

**auto** iter **=** partition\_point**(**ivec**.**begin**(),** ivec**.**end**(),** **[](int** x **)**

**{**

**return** x **%** 2 **==** 0**;** ***// tekin başladığı yeri döndürür 3'ü***

**});**

cout **<<** **\***iter **<<** '\n'**;**

**}**

**std::** **is\_sorted**

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

**{**

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

vector ivec **{** 2**,** 4**,** 6**,** 8**,** 20**};**

**auto** b **=** is\_sorted**(**ivec**.**begin**(),** ivec**.**end**());** ***// b = true***

**}**

**std::is\_sorted\_until**

***// is\_sorted\_until***

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

**{**

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

vector ivec **{** 22**,** 14**,** 6**,** 18**,** 1**,** **-**1**};**

***// büyükten küçüğe sıralamanın bozulduğu konumu verir. \*iter = 18***

**auto** iter **=** is\_sorted\_until**(**ivec**.**begin**(),** ivec**.**end**(),** greater**{});**

**}**

**heap veri yapısı**

***// heap veri yapısı***

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

**{**

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

vector**<int>** ivec**;**

rfill**(**ivec**,** 20**,** IRand**{** 0**,** 1000**});**

make\_heap**(**ivec**.**begin**(),** ivec**.**end**());** ***// max heap***

make\_heap**(**ivec**.**begin**(),** ivec**.**end**(),** greater**{});** ***// min heap***

***// heap'in en başındaki öğeyi en sona getirir çıkarabilmek için***

pop\_heap**(**ivec**.**begin**(),** ivec**.**end**());**

***// heap olma özelliğini korurs***

ivec**.**pop\_back**();** ***// sondaki öğeyi çıkarır***

***// heap'e veri ekleme***

ivec**.**push\_back**(**9021**);**

push\_heap**(**ivec**.**begin**(),** ivec**.**end**());**

**while** **(!**ivec**.**empty**())**

**{**

pop\_heap**(**ivec**.**begin**(),** ivec**.**end**());**

ivec**.**pop\_back**();**

**}**

**}**

**std::queue**

#include <queue>

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

**{**

***/\****

***-kuyruğa farklı sırada elemanları ekledik***

***-kuyruktan eleman çıkarıyoruz ama çıkan elemanlar en büyük değer***

***\*/***

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

priority\_queue**<int>** x**;**

Irand myrand**{** 0**,** 19887**};**

**for** **(int** i **=** 0**;** i **<** 20**;** **++**i**)**

**{**

**int** ival **=** myrand**();**

x**.**push**(**ival**);**

cout **<<** ival **<<** " kuyruga eklendi\n "**;**

**}**

**while** **(!**x**.**empty**())**

**{**

cout **<<** x**.**top**();**

x**.**pop**();**

**}**

**}**

**std::list**

* Ortalardan ekleme ve silme işlemleride vector'e göre daha avantajlı
* Swap işlemlerinde ve swap işlemlerir yapan algoritmalarda vector'e göre daha avantajlı
* Çok büyük verilerde list vector'e göre daha avantajlı

uyarı: profiling yapmadan list mi vector mu seçilmeli kararı verilmemeli

bidirectional\_iterator sahip bu yüzden:

* iki iteratorü büyüklük küçüklük karşılaştırması yapmak
* iki iteratorün farkını almak
* += -= gibi işlemler yapmak
* [] parantez işlemleri yapmak

bunlar syntax hatasıdır.

**std::reverse, std::remove ve std::remove\_if**

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

**{**

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

***// eğer vector olsaydı algoritma kütüphanesini kullancaktık.***

list**<**string**>** mylist**;**

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

print**(**mylist**);**

mylist**.**reverse**();** ***// listeye özel***

list**<int>** ilist**;**

rfill**(**ilist**,** 100**,** Irand**{**0**,** 5**});**

**int** ival **=** 3**;**

**auto** n **=** ilist**.**remove**(**ival**);**

list**<int>** ilist1**;**

rfill**(**ilist1**,** 100**,** Irand**{**0**,** 5**});**

**auto** n **=** ilist1**.**remove\_if**([**ival**](int** x**)** **{return** x **%** ival **==** 0**;});**

**}**

**std::** **merge**

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

**{**

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

***// karşılaştırma kriteri bilinmeli***

list**<**string**>** x**;**

list**<**string**>** y**;**

**auto** fgen **=** **[]** **{return** rname**()** **+** ' ' **+** rfname**();};**

rfill**(**x**,** 20**,** fgen**);**

rfill**(**y**,** 20**,** fgen**);**

x**.**sort**();**

y**.**sort**();**

x**.**merge**(**y**);**

cout **<<** x**.**size**()** **<<** "\n"**;** ***// 40;***

cout **<<** y**.**size**()** **<<** "\n"**;** ***// 0;***

**}**

**std::splice (bir listeden çıkartıp başka listeye ekler)**

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

**{**

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

list**<**string**>** f**{** "esin"**,** "ayse"**,** "fatma"**,** "gizem"**,** "lale"**,** "hulya"**};**

list**<**string**>** m**{** "fuat"**,** "ceyhun"**,** "tarik"**,** "cetin"**,** "kaan"**};**

f**.**splice**(**f**.**begin**(),** m**);** ***// en başa m listesini ekledik***

**}**

**std::lower\_bound, std:: upper\_bound ve std::equal\_range**

***/\****

***4 6 6 7 7 7 9 12 23 34 45 90***

***lower bound: verilen bir değer için öyle bir konum ki o değeri sırayı***

***bozmadan ekleyebileceğimiz ilk konum***

***(verilen değer 7 ise ekliyebileceğim indeks 3)***

***upper bound: verilen bir değer için öyle bir konum ki o değeri sırayı***

***bozmadan ekleyebileceğimiz son konum***

***(verilen değer 7 ise ekliyebileceğim indeks 6)***

***equal range: lower\_bound ile upper\_bound arasındaki range***

***yani 7 değeri için 3 6 arası***

***\*/***

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

**{**

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

vector**<int>** ivec**;**

***// karşılaştırma kriteri bilinmeli***

rfill**(**ivec**,** 20**,** Irand**{** 0**,** 5**});**

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

**int** key**;**

cin **>>** key**;**

**auto** iterlower **=** lower\_bound**(**ivec**.**begin**(),** ivec**.**end**(),** key**,** greater**{});**

cout **<<** "ilk ekleme yapilabilecek yerin indeksi : " **<<** distance**(**ivec**.**begin**(),** iterlower**)** **<<** '\n'**;**

**auto** iterupper **=** upper\_bound**(**ivec**.**begin**(),** ivec**.**end**(),** key**,** greater**{});**

cout **<<** "son ekleme yapilabilecek yerin indeksi : " **<<** distance**(**ivec**.**begin**(),** iterupper**)** **<<** '\n'**;**

**auto** **[**iterlower**,** iterupper**]** **=** equal\_range**(**ivec**.**begin**(),** ivec**.**end**(),** key**);**

cout **<<** key **<<** " degerinden " **<<** distance**(**iterlower**,** iterupper**)** **<<** " tane var\n"**;**

**}**

**Mülakata sorusu: bir vektöre eleman eklicez ama sıralama bozulmıcak**

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

**{**

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

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

**auto** fgen **=** **[]** **{return** rname**()** **+** ' ' **+** rfname**();** **};**

**for** **(int** i **=** 0**;** i **<** 20**;** **++**i**)**

**{**

**auto** name **=** fgen**();**

std**::**cout **<<** name **<<** "ekleniyor\n"**;**

**auto** iter **=** lower\_bound**(**svec**.**begin**(),** svec**.**end**(),** name**);**

svec**.**insert**(**iter**,** std**::**move**(**name**));**

**}**

**}**

**sorted range algorithms**

***// set\_intersection***

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

**{**

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

vector**<**string**>** x**;**

vector**<**string**>** y**;**

rfill**(**x**,** 30**,** rtown**);**

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

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

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

***// x ve y kesişimi***

list**<**string**>** slist\_inter**;**

set\_intersection**(**x**.**begin**(),** x**.**end**(),** y**.**begin**(),** y**.**end**(),** back\_inserter**(**slist\_inter**));**

***// x ve y birleşimi***

list**<**string**>** slist\_union**;**

set\_union**(**x**.**begin**(),** x**.**end**(),** y**.**begin**(),** y**.**end**(),** back\_inserter**(**slist\_union**));**

***// x fark y***

list**<**string**>** slist\_diff**;**

set\_union**(**x**.**begin**(),** x**.**end**(),** y**.**begin**(),** y**.**end**(),** back\_inserter**(**slist\_diff**));**

**}**