**2023 11 20**

**std::unordered\_Set ---equal\_to ve hasher**

***// unordered\_set için equal\_to***

**struct** Point

**{**

Point**()** **=** **default;**

Point**(double,** **double,** **double);**

**bool** **operator** **==** **(const** Point**&)const;**

**double** mx**{},** my**{},** mz**{};**

**}**

***//unordered\_set için hasher***

**template<>**

**struct** std**::**hash**<**Point**>**

**{**

std**::size\_t** **operator()(const** Point**&** pt**)const**

**{**

std**::**hash**<double>** hasher**;**

**return** hasher**(**pt**.**mx**)** **+** hasher**(**pt**.**my**)** **+** hasher**(**pt**.**mz**);**

**}**

**};**

**class** PointHasher

**{**

std**::size\_t** **operator()(const** Point**&** pt**)const**

**{**

std**::**hash**<double>** hasher**;**

**return** hasher**(**pt**.**mx**)** **+** hasher**(**pt**.**my**)** **+** hasher**(**pt**.**mz**);**

**}**

**};**

**struct** PointEqual

**{**

**public:**

**bool** **operator()(const** Point**&,** **const** Point**&)const;**

**}**

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

**{**

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

unordered\_set**<**Point**>** myset**;** ***// struct std::hash<Point>***

unordered\_set**<**Point**,** PointHasher**,** PointEqual**>** myset**;** ***// class PointHasher***

myset**.**insert**(**Point**{** 2.3**,** 5.6**,** 78.9**});**

**}**

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

**{**

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

unordered\_set**<**string**>** myset**;**

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

**{**

myset**.**insert**(**rname**());** ***// hiçbir sıra yok***

**}**

**}**

**Load Factor ve Max Load Factor**

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

**{**

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

unordered\_set**<**string**>** myset**;**

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

**{**

myset**.**insert**(**rname**());** ***// hiçbir sıra yok***

**}**

***// load factor, max load factor geçince rehash yapılır.***

std**::** cout **<<** "bucket count = " **<<** myset**.**bucket\_count**()** **<<** '\n'**;**

std**::** cout **<<** "size = " **<<** myset**.**size**()** **<<** '\n'**;**

std**::** cout **<<** "load factor = " **<<** **float(**myset**.**size**())** **/** myset**.**bucket\_count**()** **<<** '\n'**;**

std**::** cout **<<** "load factor = " **<<** myset**.**load\_factor**()** **<<** '\n'**;**

std**::** cout **<<** "max load factor = " **<<** myset**.**max\_load\_factor**())** **<<** '\n'**;**

**for** **(size\_t** i**{};** i **<** myset**.**bucket\_count**();** **++**i**)**

**{**

***// i. bucketin sizeni yazar***

std**::**cout **<<** i **<<** " " **<<** myset**.**bucket\_size**(**i**));**

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

**{**

***// bucket içindeki verileri yazar***

std**::**cout **<<** **\***iter

**}**

**}**

string name **=** "emre"**;**

**if** **(**myset**.**contains**(**name**))**

**{**

***// hangi bucket olduğunu yazar***

std**::**cout **<<** "bucket : " **<<** myset**.**bucket**(**name**)** **<<** "\n"**;**

**}**

**}**

**Function Adaptor**

Bizim callable'mızı (fonksiyon, labmda expression vb.) alıp adapte edip, özellikler verip yeni bir callable döndürür.

callable f --- > adaptor ---> callable ret

function adaptor:

* std::bind
* mem\_fn
* not\_fn
* std::invoke

**reference\_wrapper**

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

**{**

**int** x **=** 10**;**

**int** y **=** 45**;**

**int** **&**r **=** x**;**

r **=** y**;** ***// x = y***

**int** **\***p**[**20**];** ***// pointer dizisi***

**int** **&**p**[**20**];** ***// syntax hatası***

std**::**vector**<int** **\*>** myvec**;** ***// pointer container***

std**::**vector**<int&** **>** myvec**;** ***// syntax hatası;***

**}**

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

**class** ReferenceWrapper

**{**

**public:**

ReferenceWrapper**(**T **&**t**)** **:** mp**{&**t**}** **{};**

ReferenceWrapper**&** **operator=(const** T**&)**

**{**

mp **=** **&**t**;**

**}**

**operator** T**&()**

**{**

**return** **\***mp**;**

**}**

T**&** get**()**

**{**

**return** **\***mp**;**

**}**

**private:**

T **\***mp**;**

**};**

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

**void** func**(**T x**)**

**{**

**}**

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

ReferenceWrapper**<**T**>** Ref**(**T**&** x**)**

**{**

**return** ReferenceWrapper**<**T**>{**x**};**

**}**

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

**{**

**int** x **=** 12**;**

ReferenceWrapper**<int>** r**(**x**);**

cout **<<** r**;** ***// 12 cout << r.operator int& ();***

string str**(**100'000**,** 'a'**);**

func**(**ReferenceWrapper**<**string**>(**str**));** ***// func(string& str); yerine kullanabilirz***

func**(**ReferenceWrapper**(**str**));** ***// CTAD Cpp 17***

func**(**Ref**(**str**));** ***// factory method***

**}**

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

**void** foo**(**T x**)**

**{**

x **+=**100**;**

**}**

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

**{**

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

**int** ival **=** 20**;**

**int&** r **=** ival**;**

foo**(**r**);**

***// ival = 20 çünkü ref olarak gitmez template arg kurallarına göre***

cout **<<** "ival = " **<<** ival **<<**"\n"**;**

foo**<int** **&>(**r**);**

***// ival = 120***

cout **<<** "ival = " **<<** ival **<<**"\n"**;**

foo**(**reference\_wrapper**<int>{**ival**});**

***// ival = 120***

cout **<<** "ival = " **<<** ival **<<**"\n"**;**

foo**(**ref**{**ival**});**

***// ival = 120***

cout **<<** "ival = " **<<** ival **<<**"\n"**;**

**}**

**struct** BigPred

**{**

**bool** **operator()(int)**

**{**

**return** **true;**

**}**

**char** buf**[**2 **\*** 4096**]{};**

**};**

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

**{**

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

BigPred pred**;**

vector**<int>** ivec**(**100'000**);**

vector**<int>** destvec**;**

***// ref kullanarak pred nesnesini kopyalanmasını engelledik.***

copy\_if**(**ivec**.**begin**(),** ivec**.**end**(),** back\_inserter**(**destvec**),** ref**(**pred**));**

**}**

***// generate algoritması***

**int** foo**()**

**{**

**return** 12**;**

**}**

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

**{**

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

vector**<int>** ivec**(**100**);**

***// fonksiyon ya da lambda geri dönüş değerini vector'e yazar.***

generate**(**ivec**.**begin**(),** ivec**.**end**(),** foo**);**

generate**(**ivec**.**begin**(),** ivec**.**end**(),** **[]**

**{**

**return** rand**()** **\*** 2**;** ***//***

**});**

**}**

***// reference\_wrapper örnek***

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

**{**

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

mt19937 end**;**

vector**<unsigned** **int>** uvec**(**10'000**);**

std**::**cout **<<** "sizeof(eng) = " **<<** **sizeof(**eng**)** **<<** "\n"**;** ***// sizeof(eng) = 5000***

generate**(**uvec**.**begin**(),** uvec**.**end**(),** eng**);** ***// verimli degil***

generate**(**uvec**.**begin**(),** uvec**.**end**(),** ref**(**eng**));**

**}**

***// CTAD örnek***

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

**{**

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

string name **{** "necati ergin" **};**

reference\_wrapper r **=** name**;** ***// CTAD***

**}**

***// reference\_wrapper örnek***

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

**{**

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

std**::**list**<**string**>** mylist **{**"kutay"**,** "tarik"**,** "cemal"**};**

***// myvec'in elemanları listedeki elemanlarına reference***

vector**<**reference\_wrapper**<**string**>>** myvec**{**mylist**.**begin**(),** mylist**.**end**()};**

sort**(**myvec**.**begin**(),** myvec**.**end**(),** **[](auto** r1**,** **auto** r2**)**

**{**

**return** r1**.**get**()** **<** r2**.**get**();**

**});**

**for** **(const** **auto&** s **:** mylist**)**

**{**

cout **<<** s**.**get**()** **<<** " "**;**

**}**

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

**}**

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

**{**

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

**int** x **=** 321**,** y **=** 123**;**

pair p**(**ref**(**x**),** ref**(**y**));**

p**.**first **\*=** 10**;**

p**.**second **\*=** 10**;**

cout **<<** "x = " **<<** x **<<** "\n"**;** 3210

cout **<<** "y = " **<<** y **<<** "\n"**;** 1230

**}**

* reference\_wrapper incomplete type ile kullanabiliriz
* reference\_wrapper fonksiyonlarda kullanabiliriz.

**int** sumsquare**(int** x**,** **int** y**)**

**{**

**return** x **\*** x **+** y **\*** y**;**

**}**

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

**{**

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

reference\_wrapper rf **=** sumsquare**;**

**auto** val **=** rf **(**10**,** 20**);**

cout **<<** "val = " **<<** val **<<** "\n"**;**

**}**

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

**{**

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

**int** x **=** 10**,** y **=** 45**;**

reference\_wrapper r**{**x**};**

**++**r**;** ***// x = 11, y = 45***

r **=** y**;** ***// r = &y***

**++**r**;** ***// x = 11, y 46***

r**.**get**()** **=** y ***// x = y***

**}**

**cref()**

***// cref***

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

**{**

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

***// const reference\_wrapper***

**int** x **=** 356**;**

**auto** r **=** cref**(**x**)** ***//(reference\_wrapper<const int> r = x***

**}**

**std::bind**

#include <functional>

**int** foo**(int** x**,** **int** y**,** **int** z**)**

**{**

std**::**cout **<<** "x = " **<<** x **<<** "y = " **<<** y **<<** "z = " **<<** z**;**

**}**

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

**{**

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

**using** placeholders**;**

**auto** f **=** bind**(**foo**,** 10**,** 20**,** 30**);**

f**();** ***// foo(10, 20, 30); çağırır aslında***

f **=** bind**(**foo**,** 10**,** 20**,** \_1**);**

f**(**99**);** ***// foo(10, 20, 99);***

f **=** bind**(**foo**,** \_1**,** \_1**,** \_1**);**

f**(**99**);** ***// foo(99, 99, 99);***

f **=** bind**(**foo**,** \_1**,** 77**,** \_2**);**

f**(**99**,** 21**);** ***// foo(99, 77, 21);***

f **=** bind**(**foo**,** \_3**,** \_1**,** \_2**);**

f**(**10**,** 20**,** 30**);** ***// foo(30, 10, 20);***

**}**