**2024 01 03**

**std::random**

***// std::bernoulli\_distribution***

#include <random>

#include <iostream>

#include <chrono>

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

**{**

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

***// exe her çalıştığın da farklı bir seed(tohum) çalışacak***

mt19937 eng**{**std**::**chrono**::**steady\_clock**::**now**().**time\_since\_epoch**()};**

mt19937 eng1**{** random\_device**{}()** **};**

bernoulli\_distribution dist**{**0.81**}**

std**::size\_t** n **=** 1'000'000u**;**

**int** cnt**{};**

**for** **(size\_t** i**{};** i **<** n**;** **++**i**)**

**{**

**if** **(**dist**(**eng**))**

**{**

**++**cnt**;**

**}**

**}**

cout **<<** **static\_cast<double>(**cnt**)** **/** n **<<** "\n"**;**

**}**

***// her çağrıldığında aynı engine çağrılacak***

std**::**mt19937**&** engine**()**

**{**

**static** std**::**mt19937 eng**{** std**::**random\_device**{}()** **};**

**return** eng**;**

**}**

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

**{**

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

uniform\_int\_distribution dist**{**0 **,** 99**};**

cout **<<** dist**(**engine**())** **<<** "\n"**;**

**}**

***// param()***

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

**{**

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

uniform\_int\_distribution dist1**{** 0**,** 99 **};**

**auto** prm **=** dist1**.**param**();**

***//dist1 parametrelerini dist2'ye geçirdik***

uniform\_int\_distribution dist2**{**dist1**.**param**()};** ***//***

**}**

*/// algoritmalarda random*

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

**{**

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

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

mt19937 eng**;**

uniform\_int\_distribution dist**{** 4671**,** 8812**};**

generate**(**ivec**.**begin**(),** ivec**.**end**(),** **[&**eng**,** **&**dist**]{** **return** dist**(**eng**);});**

**}**

**Concurrency**

#include <thread>

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

**{**

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

***// move only type***

thread th1**{[]** **{**cout **<<** "emre bahtiyar\n"**}};** ***// ctor'u callable alır***

thread th1**{[]** **{**cout **<<** "emre bahtiyar\n"**}};** ***// ctor'u callable alır***

thread th2**{[]** **{**cout **<<** "emre bahtiyar\n"**}};** ***// ctor'u callable alır***

thread th3**{[]** **{**cout **<<** "emre bahtiyar\n"**}};** ***// ctor'u callable alır***

**}**

**void** func**(int** a**,** **int** b**,** **int** c**)**

**{**

**}**

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

**{**

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

***// eğer join() ya da detach() fonksiyonu çağrılmaz terminate fonksiyonu çağrılır***

thread tx**{** func**,** 10**,** 20**,** 30 **};**

***// fonksiyon bitene kadar bu nokta beklesin anlamına gelir***

tx**.**join**();**

***// fonksiyon ile ana thread ayrılır tx kendi biter***

tx**.**detach**();**

**}**

/\*

Bir thread nesnesi joinable ve unjoinable durumunda olabilir,

eğer bir thread nesnesi joinable durumduysa ve dtor'u çağrılırsa

doğrudan terminate fonksiyonu çağrılır.

\*/

**void** nec\_terminate**()**

**{**

std**::**cout **<<** "ne terminate cagrildi\n"**;**

std**::**abort**();**

**}**

**void** func**()**

**{**

std**::**cout **<<** "ben func fonksiyonuyum\n"**;**

**}**

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

**{**

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

set\_terminate**(**nec\_terminate**);**

**{**

thread tx**{**func**};**

tx**.**join**();** ***// bunu çağırmazsak terminate çağrılacak***

**}**

**}**

***// joinable()***

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

**{**

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

thread t**;**

cout **<<** t**.**joinable**()** **<<** "\n"**;** ***// false çünkü iş yükü yok***

thread t1**{[]** **{}};**

cout **<<** t1**.**joinable**()** **<<** "\n"**;** ***// true çünkü iş yükü var***

t1**.**join**();**

cout **<<** t**.**joinable**()** **<<** "\n"**;** ***// false artık join edilmiş***

**}**

***// thread taşıma***

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

**{**

thread t1**{[]{}};**

thread t2**{**move**(**t1**};**

cout **<<** t1**.**joinable**()** **<<** "\n"**;** ***// false artık taşındı.***

cout **<<** t2**.**joinable**()** **<<** "\n"**;** ***// true.***

t2**.**join**();**

cout **<<** t1**.**joinable**()** **<<** "\n"**;** ***// false***

cout **<<** t2**.**joinable**()** **<<** "\n"**;** ***// false.***

**}**

**void** func**()**

**{**

std**::**cout **<<** "ben func fonksiyonuyum\n"**;**

**}**

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

**{**

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

thread tx**{** func **};**

tx**.**join**();**

tx**.**join**();** ***// exception throw eder***

**}**

***// jthread --cpp 20***

**void** func**()**

**{**

**throw** std**::**runtime\_error**{** " hataaaa" **};**

**}**

**void** foo**()**

**{**

***//code***

**}**

**void** bar**()**

**{**

***/\****

***dtor çağrıldığında jthread eğer sarmaladğı thread'in joinable ise join eder***

***\*/***

jthread t**{** foo **};**

func**();** ***// exception throw eder***

t**.**join**();** ***// buraya girmez***

**}**

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

**{**

**try**

**{**

bar**();**

**}**

**catch** **(**std**::**exception **&**ex**)**

**{**

**}**

**}**

**void** func**(char** c**)**

**{**

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

**{**

std**::**cout**.**put**(**c**);**

**}**

**}**

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

**{**

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

vector**<**thread**>** tvec**(**26**);**

**for** **(auto&** t **:** tvec**)**

**{**

t **=** thread**{**func**,** c**++};** ***// taşınma semantiği***

**}**

**for** **(auto&** t **:** tvec**)**

**{**

t**.**join**();**

**}**

**}**

thread'e verdiğimiz callable exception verirse exception'ı yakalamayız

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

**{**

***// code***

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

**}**

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

**{**

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

thread tx**{** foo**,** 10**,** 56 **};**

***// foo fonksiyonun geri dönüş değerini thread nesnesi aracıyla alamayız***

tx**.**join**();**

**}**

***// thread fabrika fonksiyonu***

std**::**thread make\_thread**()**

**{**

std**::**thread tx**{[]**

**{**

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

**}};**

***// otomatik ömürlü nesneler l value to x value olr***

**return** tx**;**

**}**

std**::**thread func**(**std**::**thread tx**)**

**{**

**return** tx**;**

**}**

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

**{**

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

**auto** t1 **=** make\_thread**();**

**auto** t2 **=** move**(**t1**);**

t1 **=** func**(**move**(**t2**));**

t1**.**join**();**

**}**

***// threadlere verilebilecek callable'lar***

**class** Functor

**{**

**public:**

**void** **operator()(int** x**)const{**

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

**}**

**}**

**void** func**(int** x**)**

**{**

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

**}**

**struct** Nec

**{**

**static** **void** print**(int** x**)**

**{**

std**::**cout **<<** x**;**

**}**

**void** display**(int** x**)const**

**{**

std**::**cout **<<** x**;**

**}**

**}**

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

**{**

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

thread t1**(**func**,** 1**);** ***// fonksiyon***

thread t2**([]{int** x**}** **{**cout **<<** x **<<** "\n"**;},** 2**);** ***// lambda***

thread t3**(**Functor**{},** 3**);**

thread t4**(**Nec**::**print**,** 4**);**

Nec mynec**;**

thread t5**(&**Nec**::**display**,** mynec**,** 5**);**

***// çıktı belli olmaz***

t1**.**join**();**

t2**.**join**();**

t3**.**join**();**

t4**.**join**();**

t5**.**join**();**

**}**

**void** func**(int&** r**)**

**{**

r **+=**200**;**

**}**

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

**{**

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

**int** x **=** 67**;**

thread t**{** func**,** x**};** ***// syntax hatası***

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

thread tx**{** func**,** ref**(**x**)};** ***// syntax hatası yok;***

cout **<<** "x = " **<<** x **<<** "\n"**;** ***// 267***

**}**

**void** func**(**std**::**string **&&** r**)**

**{**

std**::**cout **<<** r**.**size**()** **<<** "\n"**;**

**auto** x **=** std**::**move**(**r**);**

**}**

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

**{**

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

string name**{** "emre bahtiyar "**};**

thread tx**{** func**,** name**};**

tx**.**join**();**

cout **<<** name**.**size**()** **<<** "\n"**;**

**}**

***// get\_id***

**void** func**()**

**{**

**}**

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

**{**

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

thread t1**{** func **};**

**auto** x **=** t1**.**get\_id**();** ***// auto --> class std::thread\_id***

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

t1**.**join**()**

**}**

/\*

tüm threadlerin id'leri farklı

eğer thread joinable durumda değilse get\_id 0 döner

\*/

***// this\_thread***

#include <syncstream>

**void** func**()**

**{**

osyncstream**{**cout**}** **<<** std**::**this\_thread**::**get\_id**()** **<<** "\n"**;**

**}**

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

**{**

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

thread t1**{**func**};**

cout **<<** t1**.**get\_id**()** **<<** "\n"**;**

t1**.**join**();**

**}**

std**::**thread**::**id main\_thread\_id**;**

**void** func**()**

**{**

**if** **(**std**::**this\_thread**::**get\_id**()** **==** main\_thread\_id**)**

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

**else**

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

**}**

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

**{**

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

main\_thread\_id **=** std**::**this\_thread**::**get\_id**();**

func**();** ***// main thread***

thread t**{** func **};**

t**.**join**();** ***// ikincil thread***

**}**

/\*

thread kütüphanesinde sonu for ve until ile biten fonksiyonlar bulunur

xxxfor duration ister

xxxuntil timepoint ister

\*/

***// sleep\_for***

**void** func**()**

**{**

std**::**this\_thread**::**sleep\_for**(**2300ms**);**

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

**}**

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

**{**

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

thread t**{**func**};**

t**.**join**();**

**}**

***// native\_handle()***

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

**{**

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

thread t**{** **[]** **{}};**

***// işletim sistemin apilerine geçicek handle verir***

**auto** handle **=** t**.**native\_handle**();**

**}**

***// hardware\_concurrency()***

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

**{**

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

***// kaç tane thread açabilirim***

**auto** b **=** thread**::**hardware\_concurrency**()**

**}**

***// thread exceptions***

**void** func**(int** x**)**

**{**

***// exception burda yakalanırsa catch bloğuna girer***

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

**{**

**throw** std**::**runtime\_error**{** "even number error\n" **};**

**}**

**}**

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

**{**

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

***/\****

***exception vermez doğrudan terminate fonksiyonu çağrılır.***

***\*/***

**try**

**{**

thread t**{** func**,** 12**};**

t**.**join**();**

**}**

**catch** **(const** std**::**exception **&**ex**)**

**{**

std**::**cout **<<** "exception caught: " **<<** ex**.**what**()** **<<** "\n"**;**

**}**

**}**

***/\****

***std::exception\_ptr***

***exception sarmalar***

***current\_exception()***

***bir exception yakalandığı zaman exception ptr nesnesi döndürür döndürür***

***rethrow\_exception()***

***exception ptr'yi rethrow eder.***

***\*/***

**void** handle\_saved\_exception**(**std**::**exception\_ptr eptr**)**

**{**

**try**

**{**

**if** **(**eptr**)**

**{**

std**::**rethrow\_exception**(**eptr**);**

**}**

**}**

**catch** **(const** std**::**out\_of\_range**&** ex**)**

**{**

std**::**cout **<<** "hata yakalandi : " **<<** ex**.**what**()** **<<** "\n"**;**

**}**

**}**

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

**{**

std**::**exception\_ptr exptr**;**

**try{**

std**::**string name**{**"emre"**};**

**auto** c **=** name**.**at**(**45**);**

**}**

**catch** **(...)** **{**

eptr **=** std**::**current\_exception**();**

**}**

**}**

std**::**exception\_ptr eptr**{** **nullptr** **};**

**void** func**(int** x**)**

**{**

std**::**cout **<<** "func cagrildi x = " **<<** x **<<** "\n"**;**

**try** **{**

**if** **(**x **%** 2 **!=** 0**)**

**{**

**throw** std**::**runtime\_error**{**"hatali arguman " **+** std**::**to\_string**(**x**)}**

**}**

**}**

**catch(...)**

**{**

eptr **=** std**::**current\_exception**();**

**}**

std**::**cout **<<** "func islevi sona eriyor\n"**;**

**}**

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

**{**

thread tx**{** func**,** 12**};**

tx**.**join**();**

**try** **{**

**if** **(**eptr**)**

rethrow\_exception**(**eptr**);**

**}**

**catch** **(const** std**::**exception**&** ex**)**

**{**

std**::**cout **<<** "exception caught: " **<<** ex**.**what**()** **<<** "\n"**;**

**}**

**}**

std**::**vector**<**std**::**exception\_ptr**>** g\_epvec**;**

std**::**mutex mtx**;**

**void** f1**()** **{throw** std**::**runtime\_error**{** "hata" **};}**

**void** f2**()** **{throw** std**::**invalid\_argument**{** "gecersiz arguman" **};}**

**void** f3**()** **{throw** std**::**out\_of\_range**{** "aralik disi değer" **};}**

**void** tfunc1**()**

**{**

**try** **{**

f1**();**

**}**

**catch** **(...)** **{**

std**::**lock\_guard guard**{**mtx**};**

g\_epvec**.**push\_back**(**std**::**current\_exception**();**

**}**

**}**

**void** tfunc2**()**

**{**

**try** **{**

f2**();**

**}**

**catch** **(...)** **{**

g\_epvec**.**push\_back**(**std**::**current\_exception**();**

**}**

**}**

**void** tfunc3**()**

**{**

**try** **{**

f3**();**

**}**

**catch** **(...)** **{**

g\_epvec**.**push\_back**(**std**::**current\_exception**();**

**}**

**}**

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

**{**

**using** **namespace** sd**;**

thread t1 **{**tfunc1**};**

thread t2 **{**tfunc2**};**

thread t3 **{**tfunc3**};**

t1**.**join**();**

t2**.**join**();**

t3**.**join**();**

**for** **(const** **auto** **&**ex **:** g\_epvec**)**

**{**

**try** **{**

rethrow\_exception**(**ex**);**

**}**

**catch** **(const** std**::**exception**&** ex**)** **{**

std**::**cout **<<** "exception caught : " **<<** ex**.**what**()** **<<** "\n"**;**

**}**

**}**

**}**

***// make\_exception\_ptr()***

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

**{**

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

exception\_ptr eptr**;**

**try**

**{**

**throw** std**::**runtime\_error**{** "hata" **};**

**}**

**catch(...)**

**{**

eptr **=** current\_exception**();**

**}**

***// yukarıdaki yerine***

**auto** eptr **=** make\_exception\_ptr**(**std**::**runtime\_error**{** "hata" **};**

**}**

**thread\_local storage**

***// thread\_local bir keyword***

**thread\_local** **int** x**{**0**};**

**void** foo**(const** std**::**string**&** thread\_name**)**

**{**

***// her thread'in thread\_local değişkeni o thread için tek***

**++**x**;**

osyncstream**{**cout**}** **<** **<** "thread" **<<** thread\_name **<<** " x = " **<<** **<<** "\n"**;**

**}**

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

**{**

jthread tx**{** foo**,** "tx" **};** ***// thread tx x = 1***

jthread ty**{** foo**,** "ty" **};** ***// thread tx x = 2***

jthread tm**{** foo**,** "tm" **};** ***// thread tx x = 3***

jthread tz**{** foo**,** "tz" **};** ***// thread tx x = 4***

**}**

**thread\_local** **int** ival **=** 0**;**

**void** thread\_func**(int** **\***p**)**

**{**

**\***p **=** 42**;**

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

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

**}**

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

**{**

std**::**cout **<<** "ival = " **<<** ival **<<** "\n"**;** ***// ival = 0***

ival **=** 9**;**

std**::**cout **<<** "ival = " **<<** ival **<<** "\n"**;** ***// ival = 9***

std**::**thread t**{**thread\_func**,** **&**ival**};**

t**.**join**()** ***// \*p = 42, ival = 0***

std**::**cout **<<** "ival = " **<<** "\n"**;** ***// ival = 42***

***/\****

***-thread\_func'ta ival hala 0 thread\_local olduğu için***

***-main thread ival'nin adresini gönderdik o 9 değerini 42 yaptı***

***bundan dolayı ival son olarak 42 kaldı***

***\*/***

**}**

***// ÖRNEK***

std**::**mutex mtx**;**

**void** foo**(int** id**)**

**{**

**int** x **=** 0**;** ***// automatic storage***

**static** **int** y**=** 0**;** ***// static storage***

**thread\_local** **int** z **=** 0**;** ***// thread\_local storage***

**++**x**;**

**++**z**;**

mtx**.**lock**();**

**++**y**;**

std**::**cout **<<** "thread id: " **<<** id **<<** " x(automatic) = " **<<** x **<<** "\n"**;**

std**::**cout **<<** "thread id: " **<<** id **<<** " y(static) = " **<<** y **<<** "\n"**;**

std**::**cout **<<** "thread id: " **<<** id **<<** " z(thread\_local) = " **<<** z **<<** "\n"**;**

mtx**.**unlock**();**

**}**

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

**{**

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

jthread t1**{** foo**,** 1**};**

jthread t2**{** foo**,** 2**};**

jthread t3**{** foo**,** 3**};**

jthread t4**{** foo**,** 4**};**

***/\****

***static değişken 4 e kadar çıkar***

***automatic ve thread\_local 1 çıkar hep***

***\*/***

**}**

***// ÖRNEK***

**thread\_local** std**::**string name**{** "cemal" **};**

**void** func**(const** std**::**string**&** surname**)**

**{**

name **+=** surname**;**

***// name'lerin adresleri farklı olur.***

std**::**osyncstream**{** std**::**cout **}** **<<** name **<<** "&name = " **<<** **&**name **<<** "\n"**;**

**}**

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

**{**

**const** **char\*** **const** pa**[]** **=** **{** "akkan"**,** "toprak"**,** "bahtiyar"**,** "ersoy"**,** "canberk"**};**

vector**<**thread**>** tvec**;**

**for** **(auto** p **:** pa**)**

**{**

tvec**.**emplace\_back**(**func**,** p**);**

**}**

**for** **(auto&** t **:** tvec**)**

t**.**join**();**

**}**

**class** Myclass **{**

**public:**

Myclass**(int** x**)**

**{**

std**::**osyncstream**{** std**::**cout **}** **<<** "Myclass ctor x = " **<<** x **<<** " this = " **<<** **this** **<<** "\n"**;**

**}**

**~**Myclass**()**

**{**

std**::**osyncstream**{** std**::**cout **}** **<<** "Myclass dtor this = " **<<** **this** **<<** "\n"**;**

**}**

**};**

**void** func**(int** x**)**

**{**

**thread\_local** Myclass m**{** x **};**

**}**

**void** foo**(int** x**)**

**{**

std**::**osyncstream**{** std**::**cout **}** **<<** "foo(int) cagrildi x = " **<<** x **<<** "\n"**;**

func**(**x**);**

std**::**osyncstream**{** std**::**cout **}** **<<** "foo(int) sona eriyor x = " **<<** x **<<** "\n"**;**

**}**

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

**{**

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

**{**

jthread t1**{** foo**,** 1**};**

jthread t2**{** foo**,** 1**};**

jthread t3**{** foo**,** 1**};**

jthread t4**{** foo**,** 1**};**

**}**

std**::**cout **<<** "main devam ediyor\n"**;**

**}**

***// ÖRNEK***

/\*

thread\_local değişken yerel değişken de olabilir. Böyle bir değişkenin hayatı

kapsamı sonununda bitmez. Threadin yürütülmesi sonunda hayatı sona erer.

\*/

**thread\_local** std**::**mt19937 eng**{** 4542345u**};**

**void** foo**()**

**{**

std**::**uniform\_int\_distribution dist**{** 10**,** 99**};**

**for** **(int** i **=** 0**;** i**<** 10**;** **++**i**)**

**{**

std**::**cout **<<** dist**(**eng**)** **<<** ' '**;**

**}**

**}**

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

**{**

std**::**thread t1**{** foo **};**

t1**.**join**();**

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

std**::**thread t2**{** foo **};**

t2**.**join**();**

**}**

***//ÖRNEK***

***// Myclass nesnesi thread\_local olduğu için thread'in sonuna kadar yaşar***

**class** Myclass

**{**

**public:**

Myclass**()**

**{**

std**::**osyncstream**{** std**::**cout **}** **<<** "Myclass ctor this : " **<<** **this** **<<** "\n"**;**

**}**

**~**Myclass**()**

**{**

std**::**osyncstream**{** std**::**cout **}** **<<** "Myclass dtor this : " **<<** **this** **<<** "\n"**;**

**}**

**}**

**void** foo**()**

**{**

std**::**ostringstream**{** std**::**cout **}** **<<** "foo called\n"**;**

**thread\_local** Myclass m**;**

std**::**ostringstream**{** std**::**cout **}** **<<** "foo ends\n"**;**

**}**

**void** bar**()**

**{**

**using** **namespace** std**::**chrono\_literals**;**

std**::**ostringstream**{** std**::**cout **}** **<<** "bar called\n"**;**

foo**();**

std**::**this\_thread**::**sleep\_for**(**3s**);**

std**::**ostringstream**{**std**::**cout**}** **<<** "bar ends\n"**;**

**}**

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

**{**

std**::**thread t**{** bar **};**

t**.**join**();**

**}**

/\*

data race: birden fazla thread paylaşımlı değişkeni kullancak ve

en az biri yazma amaçlı kullancak. data race varsa tanımsız davranıştır

race condition: paylaşımlı değişkenin kullanılması ama

herhangi bir tanımsız davranış olmak zorunda değil

\*/

***// data race***

**int** gcount **=** 0**;**

**void** foo**()**

**{**

**for** **(int** i **=** 0**;** i **<** 100'000**;** **++**i**)**

**++**gcount**;**

**}**

**void** bar**()**

**{**

**for** **(int** i **=** 0**;** i **<** 100'000 **<<** **++**i**)**

**--**gcount**;**

**}**

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

**{**

***/\****

***thread kullanmasak önce gcount 100'000 olacak sonra 0 olacak***

***data race olmıcak bu durumda.***

***thread kullanınca gcount son değeri belirsiz olacak bu data race***

***tanımsız davranış***

***\*/***

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

**{**

jthread t1**{** foo **};**

jthread t2**{** bar **};**

**}**

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

**}**

***// mutual exclusion***

***/\****

***acquire***

***critical section***

***release***

***\*/***

***// std::mutex***

**int** gcount **=** 0**;**

std**::**mutex mtx**;**

**void** func**()**

**{**

mtx**.**lock**();** ***// kilit boştaysa buraya gelen thread kilidi edenir. Diğer thread'ler giremez***

**++**gcount**;**

mtx**.**unlock**();** ***// kilit serbest bırakılr***

**}**

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

**{**

thread t1**{** func **};**

thread t2**{** func **};**

thread t3**{** func **};**

thread t4**{** func **};**

**}**