**2023 12 20**

**Tie**

***// tie***

#include <iostream>

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

**{**

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

cin**.**tie**();** ***// ostream\* döner***

**if** **(auto** osptr **=** cin**.**tie**())**

**{**

***// cin cout nesnesi tie edilmiş yani bağlanmış***

cout **<<** "cin is tied. the address of the stream cin tied to is : " **<<** osptr **<<** "\n"**;**

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

**}**

**}**

**lambda ifadeleri**

***// lambda init capture***

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

**{**

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

**int** x **=** 46**;**

***// derleyici a'nın değerine x ifadesinin değerine eşitler***

**auto** fn **=** **[**a **=** x**]()** ***// a closure type'taki veri elemanına verdiğim isim***

**{**

**return** a **\*** 5**;**

**};**

***// genel kullanım böyle***

**auto** fn **=** **[**x **=** x **+** 5**]()**

**{**

**return** x**;**

**};**

**}**

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

**{**

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

**auto** uptr **=** make\_unique**<**string**>(**"necati ergin"**);**

**auto** f **=** **[&**uptr**]()**

**{**

cout **<<** **\***uptr **<<** "\n"**;**

**};**

f**();**

cout **<<** **(**uptr **?** "dolu" **:** "bos"**)** **<<** "\n"**;** ***// dolu çünkü adresini gönderdik***

***// lambda ifadesine biz taşıma yapmak istersek eğer:***

**auto** f **=** **[**uptr **=** move**(**uptr**)]()**

**{**

cout **<<** **\***uptr **<<** "\n"**;**

**};**

cout **<<** **(**uptr **?** "dolu" **:** "bos"**)** **<<** "\n"**;** ***// boş çünkü taşıdık.***

**}**

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

**{**

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

**int** x **=** 5**;**

**auto** fn **=** **[&**x **=** x**]** **{++**x**;};**

fn**();**

cout **<<** "x = " **<<** x **<<** "\n"**;** ***// x = 6***

fn**();**

fn**();**

cout **<<** "x = " **<<** x **<<** "\n"**;** ***// x = 8***

**}**

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

**{**

**int** a**[**10**]{};**

**auto** fn **=** **[**a**]** **{**

std**::**cout **<<** **typeid(**a**).**name**()** **<<** "\n"**;**

**};**

fn**();** ***// int const [10]***

**auto** fn1 **=** **[**a **=** a**]** **{**

std**::**cout **<<** **typeid(**a**).**name**()** **<<** "\n"**;**

**};**

fn**();** ***// int \****

**}**

***// this lambda capture***

**class** Nec

**{**

**public:**

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

**{**

**auto** fn **=** **[](int** a**)** **{return** a **\*** a**;};**

**auto** fn1 **=** **[**x**](int** a**)** **{return** a **\*** x**;};**

***// copy capture ama bir pointer olduğu için reference etmem ile fark yok***

**auto** fn2 **=** **[this](int** a**)** **{return** a **\*** **(**mx **+** my**;};**

**auto** fn2 **=** **[&](int** a**)** **{return** a **\*** **(**mx **+** my**;};**

**}**

**private:**

**int** mx**,** my**;**

**}**

**std::ratio**

***// std::ratio***

**template** **<int** NUM**,** **int** DEN **=** 1**>**

**class** Ratio**{**

**public:**

**constexpr** **static** **int** num **=** NUM**;**

**constexpr** **static** **int** den **=** DEN**;**

**};**

#include <ratio>

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

**{**

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

***// 2 / 5;***

ratio**<**2**,** 5**>::**num**;** ***// 2***

ratio**<**5**,** 45**>::**num**;** ***// 1***

ratio**<**5**,** 45**>::**den ***// 9***

ratio**<**12**,** **-**18**>::**num ***// -2***

ratio**<**12**,** **-**18**>::**den ***// 3***

cout **<<** **typeid(**ratio**<**2**,** 5**>::**type**).**name**()** **<<** "\n"**;** ***// struct std::ratio<2,5>***

**}**

***// ratio meta fonksiyonları***

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

**{**

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

**using** rt **=** ratio\_add**<**ratio**<**1**,** 3**>,** ratio**<**2**,** 5**>>::**type**;** ***// 11 / 15***

rt**::**num**;** ***// 11***

rt**::**den ***// 15***

ratio\_multiply**<**ratio**<**2 **,**3**>,** ratio**<**3**,** 2**>>**

ratio\_equal**<**ratio**<**1**,** 3**>,** ratio**<**2**,** 6**>>::**value ***// true***

ratio\_less**<**ratio**<**87123**,** 98134**>,** ratio**<**87981**,** 99134**>::**value ***// false***

milli**::**num ***// 1***

milli**::**den ***// 1000***

**}**

**std::chrono**

***// duration türü (süreyi ifade eden bir tür)***

#include <chrono>

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

**{**

std**::**chrono**::**duration

**namespace** chr **=** std**::**chrono**;** ***// namespace alias***

chr**::**duration**;**

**}**

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

**using** **namespace** chrono**;**

**using** HalfDay **=** std**::**chrono**::**duration**<int,** std**::**ratio**<**12 **\*** 60 **\*** 60**>>;**

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

**{**

***// duration default açılım***

duration**<int,** ratio**<**1**,** 1**>>;**

duration**<int,** ratio**<**1**>>;**

duration**<int>;**

duration**<int,** ratio**<**1**,** 2**>>** ***// yarım saniyeyi belirtir (1, 2)***

duration**<int,** milli**>** ***// milli saniye belirtir***

duration**<int,** ratio**<**3600**>>** ***// saatleri***

**}**

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

**{**

duration**<int>** x **=** 12**;** ***// copy init syntax hatası***

duration**<int>** x**(**12**)** ***// direct init***

duration**<int>** x**{**12**};**

**}**