lessons 55 - 59

DERS 55

KONU: std::cout / cin

cout << endl; // endl nasıl çalışır

```
class ostream{
   ostream& operator<<(const char*);
   ostream& operator<<(int);
   // ... others
   ostream& operator<<( ostream&(*fp)(ostream&) )  // fp alıo BU
   {
      return fp(*this);  // BÜYÜK HİLE :) @ftg185
   }
}

ostream& endl(ostream&)  // global function, @ftg185 sayesinde c
{
   // codes...
}

ostream& my_free_function(ostream& OS)  // global function
{
   // codes...
   return OS;
}

int main(){

   std::Cout << "hello" << endl << my_free_function << endl;   /

}
```

DERS 56

KONU: std::cout / cin devam

DERS 57

KONU: fstream

DERS 58

const kullanmak çok önemli , c++ de daha kolay

```
const int ix;
c = a * b
if(a > 854){
   c/2;
}
// hesaplar felan ....
ix = c;  // HATA bu yüzden const yapamıoduk C de ama C++ öle değ

// ----------------------------------------
// IIFE (Immediately Invoked Function Expression)   - aşağıdaki
const int ix = [&](){           // HEP BÖYLE YAP ARTIK ve CONST
                    c = a * b
                    if(a > 854){
                       c/2;
                        }
           return c;
         }();
```

VOL1: C de ki alışkanlıkları bırak: Örnek, out parametre

```
void func(std::vector<string> & data);  // böle değil
std::vector<string> func( );  // böyle yap OUT parametreyi
```

VOL2: her excaptionı yakalamayın. yakalayınca ne iş yapacağını bildiğin excaptını yakala sadece

VOL3: using ve tür eş ismi sık kullan okunabilirlik için

VOL4: x.cpp scopu içinde static function tanımlayacağına adı olmayan namespace kullanabilirsin.

```
static void check_is_true()[}
static int x;

// Bu yukardaki yerine aşağıdakini kullanabilirsin !!!

namespace {
  void check_is_true(){}
  int x;
}

void test::func(){
   check_is_true();
   x++;
}
```

DERS 59

lambda init capture(LAMBDA) - C++ 14

```
int a = 5;

auto f = [x = a + 5](){   // a nın türünü derleyici bulur. aslı
      return x * 2;   // yani x artık (a+5),  UNUTMA , burda x 
   };

f();
```

Buna neden ihtiyacımız olsun?

```
int main(){
   auto uptr = make_unique<string>("neco");

   auto f = [&uptr](){    // unique copy e karşı kapalı olduğun
      cout << *uptr;
   };
```

```
    f();
    cout << (uptr ? "dolu" : "bos");   // CEVAP: dolu , ama ben

  auto xx = move(uptr);    // uptr kaynağını verdi artık boş, yu
}

// çözüm :
 auto f = [xptr = move(uptr)](){    // ÇÖZÜM BU , not: xptr değ
    cout << *xptr;
 };
```

```
int x = 5;

auto fn = [ &y = x]{y++; }; // artık y adres tutuo yani x in adı

fn();
fn();
std::cout  << x << "\n";  // CEVAP : 7
```

this capture(LAMBDA): soru şu class memberlarını nasıl lambda içine geçeriz, lambda free yani global class yazıo, bu durumu değişmek yani derleyicinin lamda yerine yazdığı global classı , çağırdığımız ilgili classın içine o lambda classını ekletmek lazım. Olay bu.

```
class my_class{
    void foo(){
         auto fn = [this](){   return x*2;  }  // artık derleyic
// veya
         auto fn = [&](){   return x*2;  }   // bu ikiside çözü

     auto fn = [=](){   return x*2;  }  // bu farklı gördük bunu
       auto fn = [=, this](){   return x*2;  } // bir yukardaki
}
```

```
    int x;
}
```

`std::ratio`, C++11 , Bu sınıf, rasyonel sayıları temsil etmek için kullanılır. Rasyonel sayılar, tam sayılar arasında oranları temsil etmek için kullanılan özel bir sayı türüdür. Mesela 2 / 5 tutuo. CONSTEXPR tabiki.

`std::chrono`, C++11. Bu başlık dosyası, zamanı ölçmek ve işlemek için standart kütüphanenin bir parçasıdır.

clock : — — — — (t1) — — — — (t2) — — — — > time

time_point (t1 ve t2) : tarih zamanda bir nokta demek. 2 time point bir birinden çıkarsa bir süre yani **duration (t2 - t1 farkı yani)** elde ederiz. ve tabi t1 + t2 time_pointlerin toplanması anlamsız.

```
std::chrono::duration<int>>  // Bu tür tick saniyede 1 artı deme
std::chrono::duration<int , ratio<1, 2>>  // Bu tür yarım saniye
std::chrono::duration<int , ratio<1, 4>>  // Bu tür her bir bir
std::chrono::duration<int , ratio<1, 1000>>  // Bu tür her bir b
std::chrono::duration<int , milli> // = std::chrono::duration<in
std::chrono::duration<int , ratio<3600, 1>> // saatde bir
```