

Chrono Kütüphanesi: (Bu kütüphane app 20 ye kadar)

→ C'deki time.h kütüphanesi gibi

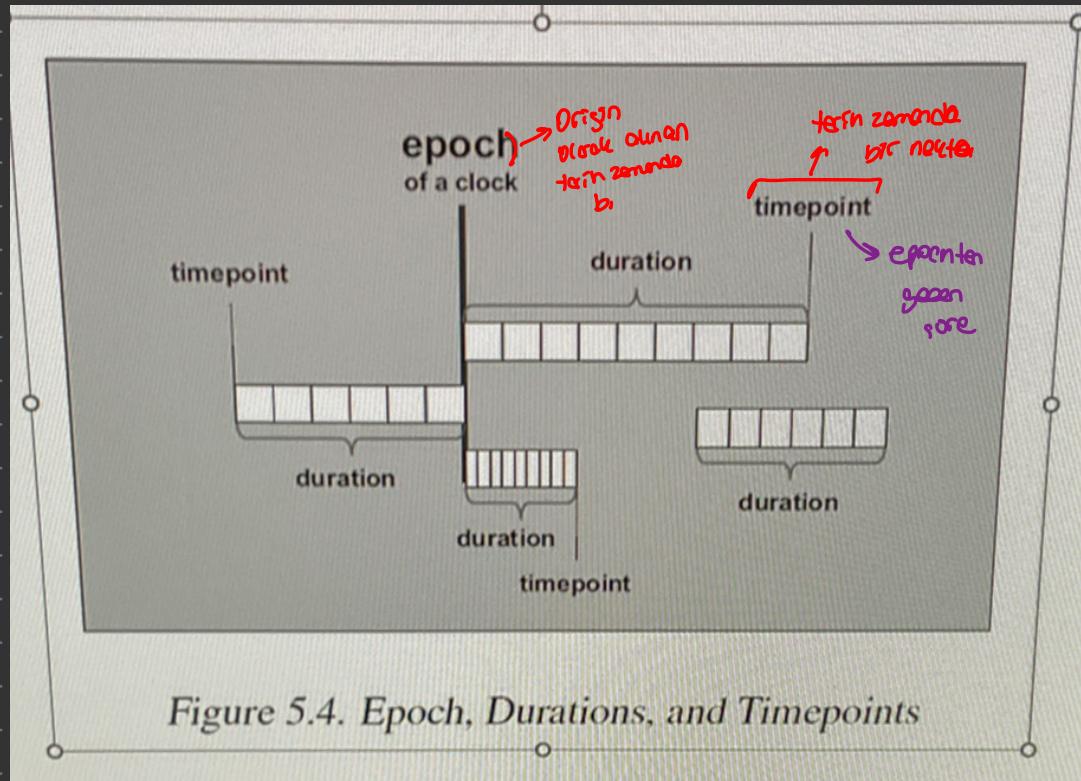


Figure 5.4. Epoch, Durations, and Timepoints

```
int main()
{
    std::chrono::duration<> I
```

→ Duration bir template ve 2 tane parametresi var. (tick time, tick time type)

→ Neden farklıdır? Çünkü zamanın ölçüsü. std::chrono:: seconds enderdir.

```
int main()
{
    using namespace std;
    using namespace chrono;

    duration<int, ratio<1, 1>>
```

tick time tick time type

ticiklerin
her biri
saniye
ve
kaç tık var onu
sayan

```
int main()
{
    using namespace std;
    using namespace chrono;

    using Day = duration<int, ratio<60 * 60 * 24, 1>>
```

rep period

ratio<intmax_t_Nx, intmax_t_Dx = 1/64>

bir gün saniye de
oldu.
ratio'nun default argümanı
1

* Lowest Implementasyonlar:

```
using namespace std;
using namespace chrono;

cout << typeid(seconds).name() << '\n';
cout << typeid(milliseconds).name() << '\n';
cout << typeid(microseconds).name() << '\n';
cout << typeid(nanoseconds).name() << '\n';

+ minutes
+ hours do vs.
```

* BT Sıra Durdurma Tırak Değerlerini:

```
using namespace std;
using namespace chrono;
```

```
milliseconds ms = 234;
```

Copy initialization kullanılmış!

Default initialization ve braces initialization yapılmış!

* Eger atama yapılmazsa, time duration atanabilir.

```
int main()
{
    using namespace std;
    using namespace chrono;
```

```
constexpr auto nsec = 3456ns;
    ns son ekz
    getirerek atanır
```

```
auto x1 = 456s; user
auto x2 = 456ms; defined
auto x3 = 456us; literals
auto x4 = 456ns;
auto x5 = 456min;
```

* Sağlıcık duration'a, durationlar da sağlayarak implicit dönüşüm!

```
auto x = 456s; Bu türkçe
                türkçe
                sağlıcık
                değer
int ival = x.count()
```

* Fine / Coarse Dönüşüm:

Küçük türden büyük tür → dönüştürken veri kaybı olmaz. (BÝ: 35 saniyeden nanosekize → örtük dönüşüm)

Büyük türden küçük tür → dönüştürken veri kaybı olabilir. → farklı dönüşümler!

```
int main()
{
    using namespace std;
    using namespace chrono;

    nanoseconds nsec = 456ms;
    cout << nsec.count() << "\n";
    → Dönüşüm örtük
        olacak yaptı
```

```
int main()
{
    using namespace std;
    using namespace chrono;

    milliseconds msec = 456ns;
    cout << msec.count() << "\n";
    syntax error.
```

```

int main()
{
    using namespace std;
    using namespace chrono;

    auto nan_sec = 456ns;
    milliseconds msec = duration_cast<milliseconds>(nan_sec);
    Küçük birlikteki kullanılmış operator.

```

→ msec ve msec.count() işlevleri var. count() kullanılabilir.

```

int main()
{
    using namespace std;
    using namespace chrono;

    milliseconds ms{ 987991 };
    auto x = duration_cast<seconds>(ms);

    cout << x << '\n'; → 987 print ediyor.

```

```

int main()
{
    using namespace std;
    using namespace chrono;

    constexpr auto dur = 456s + 2345ms + 512345us + 1234761ns;
    Durumun toplam nanosekilde olur

```

* Generic Print Duration:

```

template <typename Rep, typename Period>
std::ostream& print_duration(std::ostream& os, const std::chrono::duration<Rep, Period>& dr)
{
    return os << dr.count() << " * " << Period::num << '/' << Period::den;
}

int main()
{
    using namespace std::literals;

    print_duration(&os::std::cout, dr:753min);

```

* Chrono Sınıflarında Kesinleştirme İşlemleri:

```
int main()
{
    using namespace std;
    using namespace std::chrono;

    long long int msx, nsx;

    cout << "milisaniye olarak sureyi girin : ";
    cin >> msx;
    cout << "nanosaniye olarak sureyi girin : ";
    cin >> nsx;
    cout << boolalpha;

    cout << (milliseconds{ msx } > nanoseconds{ nsx }) << '\n';
```

(const char [34]) "nanosaniye olarak sureyi girin : "
Search Online

* Increment / Decrement:

```
int main()
{
    using namespace std;
    using namespace std::chrono;

    auto sec = 345s;

    for (int i = 0; i < 100; ++i) {
        cout << sec++ << '\n';
    }
}
```

+ ve --操作器 menzili
yapılabilir. Operator overloading var!

Utilities library Date and time utilities std::chrono::duration

std::chrono::duration<Rep, Period>::operator+=, -=, *=, /=, %=

duration& operator+=(const duration& d);	(1)	(until C++17)
constexpr duration& operator+=(const duration& d);		(since C++17)
duration& operator-=(const duration& d);	(2)	(until C++17)
constexpr duration& operator-=(const duration& d);		(since C++17)
duration& operator*=(const rep& rhs);	(3)	(until C++17)
constexpr duration& operator*=(const rep& rhs);		(since C++17)
duration& operator/=(const rep& rhs);	(4)	(until C++17)
constexpr duration& operator/=(const rep& rhs);		(since C++17)
duration& operator%=(const rep& rhs);	(5)	(until C++17)
constexpr duration& operator%=(const rep& rhs);		(since C++17)
duration& operator%=(const duration& rhs);	(6)	(until C++17)
constexpr duration& operator%=(const duration& rhs);		(since C++17)

Performs compound assignments between two durations with the same period or between a duration and a tick value.

If rep_ is the member variable holding the number of ticks in this duration object,

- 1) Equivalent to rep_ += d.count(); return *this;
- 2) Equivalent to rep_ -= d.count(); return *this;
- 3) Equivalent to rep_ *= rhs; return *this;
- 4) Equivalent to rep_ /= rhs; return *this;
- 5) Equivalent to rep_ %= rhs; return *this;
- 6) Equivalent to rep_ %= d.count(); return *this;

* Clock Ticks:

- System Clock:

- Brum. system softim2
- Asynchrone. (Aysynchroner saat = Steady Clock)
- now Ankişunu time point oluşturur. (Günüt time point, duration a bg.)

```
int main()
{
    using namespace std;
    using namespace chrono;
    auto tp = system_clock::now();
```

time point.

```
int main()
{
    using namespace std;
    using namespace chrono;

    auto tp1 = system_clock::now();
    auto tp2 = system_clock::now();
    tp2 - tp1
```

→ Aradın gelen süre

Bu Problem
sorun
öleteli.

```
int main()
{
    using namespace std;
    using namespace chrono;

    vector<int> ivec;
    mt19937 eng;
    uniform_int_distribution dist{ 0, 1'000'000 };

    auto tp_start = steady_clock::now();
    generate_n(back_inserter(ivec), 1'000'000, [&]() {return dist(eng); });
    sort(ivec.begin(), ivec.end());
    auto tp_end = steady_clock::now();

    cout << (tp_end - tp_start) << '\n';
```

CPP 20 oldugu için
count kullanımda!! 20 degeri kullan!!

* Epochen Gecen Sureler Bulma:

```
int main()
{
    using namespace std;
    using namespace chrono;
    cout << duration<double, ratio<60 * 60 * 24>>{system_clock::now().time_since_epoch().count();}
```

gon

I epochen geçen süre

* to_time_t:

```
int main() → include <ctime>
{
    using namespace std::literals;

    using Time = std::chrono::system_clock;
    time_t time_point_to_time_t
    auto timer = Time::to_time_t(Time::now() + 782'364s);
    std::cout << std::ctime(&timer) << '\n';
```

* from_time_t: \rightarrow time_t to tmepoint

* Ornek:

```
1 int main()
2 {
3     using namespace std;
4
5     auto std::chrono::system_clock::time_point tp_now = system_clock::now();
6
7     print_time(timer:system_clock::to_time_t(_Time:tp_now));
8     int min;
9     cout << "kac dakika : ";
10    cin >> min;
11    time_t timer = system_clock::to_time_t(_Time:tp_now - minutes{ min });
12    cout << min << " dakika oncesi : ";
13    print_time(timer);
14    timer = system_clock::to_time_t(_Time:tp_now + minutes{ min });
15    cout << min << " dakika sonrasi : ";
16    print_time(timer);
17 }
```

* Vocabulary Types: \rightarrow Cpp 17 ile eklenen

- std::optional
- std::variant
- std::any \rightarrow class

* Optional:
- Degirmenin degeri varsa degerini, yoksa deger almadiği standır.
- return degeri döndür / fonksiyon parametresi olabilir / class member olabilir.

```
1 #include <optional>
2 #include <iostream>
3 #include <string>
4 #include "date.h"
5
6
7 int main()
8 {
9     using namespace std;
10    optional<Date> x; Deger ya Date
11 } ya da yok.
```

- has_value ve operator bool ile degerinin olup olmadığını sorulabilir

```
1 using namespace std;
2
3 int main()
4 {
5     optional<string> x; default init oldugu icin degeri yok.
6
7     cout << boolalpha;
8
9     cout << x.has_value() << "\n"; false
```

```
1 optional<string> x;
2
3
4 if (x) {
5     std::cout << "degere sahip\n";
6 }
7 else {
8     std::cout << "degere sahip degil\n";
9 }
```

- Dereference ederek sonucunu okuyarak değer get edebiliriz. Fakat eğer değer yoksa ve deref edersek : UNDEFINED BEHAVIOR !!

→ sentetik hatalı vermen!

```
using namespace std;

int main()
{
    optional<int> x;

    /**
     * if (x)
     *     cout << *x << '\n'; */
}
```

önre değer var mı
yok mu test edip
daha sonra deref
etmeye en sağlamış.