## * String Length:

→ const member fonksiyondur.

→ dönüş size_t / auto ile

```
int main()
{
    std::string s1;
    std::string s2{};

    //auto len = s1.length();
    //std::size_t len = s1.length();
    //std::string::size_type len = s1.length();
}
```

• Default initialization olduğu için length = 0

→ Fakat: String class aynı zamanda bir container'dır. Container'ların genel bir fonksiyonu → size const üye fonksiyonu vardır.
→ length ile aynı
→ generic programlamada size kullan

• Empty: • Kaçıklar var mı yok mu, sınamasını yapan ve boolean döndüren member fonksiyon

• Clear: • Diğer containerlarda da var. String'in içini siler

• Capacity: • Realloc Etmeden! alabileceği öğe sayısı.

- Hatırlatma: 1) String için 3 pointer yeterli → start
→ push back
→ capacity

2) Otomatik ömür → stack'te
Dinamik ömür → heap
Global → Data

• Reserve: • String kapasitesini reserve eder.

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

    string str{ "13 haziran 2022 pazartesi" };→ 1. alloc
    str.reserve(7'869'234);→ dışı sağlandı. → 2. kez alloc

    auto cap = str.capacity();
    int cnt{};

    while (true) {
        str.push_back('x');
        if (str.capacity() > cap) {
            std::cout << ++cnt << " size = " << str.size() << " cap = " << str.capacity() << "\n";
            cap = str.capacity();
            (void)getchar();
        }
    }

}
```

- **Erase:** • Verilen sayıdan itibaren siler

- **Shrink to Fit:** • Size/Vice, allocated olanı da kaldırır.

## * String Sınıfı Constructorları:

- Char parametreli string constructor yok. `std::string s1('A')` → syntax hatası
- Tek parametreli yapmak için: CString constructor
  Fill constructor
  Initializer list constructor.

```
std::string s1("A");  → CString
std::string s2(1, 'B'); → Fill
std::string s3{ 'C' }; → init list
```
→ Aradaki farkı unutma
init list {3}'te sayı verme

```
std::cout << "s1 = " << s1 << "\n";
std::cout << "s2 = " << s2 << "\n";
std::cout << "s3 = " << s3 << "\n";
```

### → Substring Constructors:

```
int main()
{
    std::string s1{ "volkan gundogdu" };
    std::string s2{ s1, 7 };
    std::string s3{ s1, 7, 3 };

    std::cout << "[" << s2 << "]\n";
    std::cout << "[" << s3 << "]\n";
```

### → Özet:

```
default (1)      string();
copy (2)         string (const string& str);
substring (3)    string (const string& str, size_t pos, size_t len = npos);
from c-string (4) string (const char* s);
from buffer (5)  string (const char* s, size_t n);
fill (6)         string (size_t n, char c);
range (7)        template <class InputIterator>  string  (InputIterator first, InputIterator last);
initializer list (8) string (initializer_list<char> il);
move (9)         string (string&& str) noexcept;
```

## Construct string object

Constructs a string object, initializing its value depending on the constructor version used:

**(1) empty string constructor (default constructor)**
Constructs an empty string, with a length of zero characters.

**(2) copy constructor**
Constructs a copy of ***str***.

**(3) substring constructor**
Copies the portion of ***str*** that begins at the character position ***pos*** and spans ***len*** characters (or until the end of ***str***, if either ***str*** is too short or if ***len*** is string::npos).

**(4) from c-string**
Copies the null-terminated character sequence (C-string) pointed by ***s***.

**(5) from buffer**
Copies the first ***n*** characters from the array of characters pointed by ***s***.

**(6) fill constructor**
Fills the string with ***n*** consecutive copies of character ***c***.

**(7) range constructor**
Copies the sequence of characters in the range `[first,last)`, in the same order.

**(8) initializer list**
Copies each of the characters in ***il***, in the same order.

**(9) move constructor**
Acquires the contents of ***str***.
***str*** is left in an unspecified but valid state.

## * String Arama Fonksiyonları:

***npos:** • Const expr static bir veri elemanı

• Iki önemli özelliği var: Arama fonksiyonlarında kullanılır. → eğer hatalar kullanılmasın arama işlemleri için string sınıfının kendi global arama fonksiyonları kullanılır.

• Değeri size_t türünden alabileceği en büyük değer

```
string str{ "eray goksu" };

auto idx = str.find('t');
if (idx != string::npos) {
}
```

→ eğer varsa türünden kullanım.

• Begin ve End:

• Range based for loop:

```
string str{ "eray goksu" };

for (auto iter = str.begin(); iter != str.end(); ++iter) {
}
```

* iter → konuma erişir.

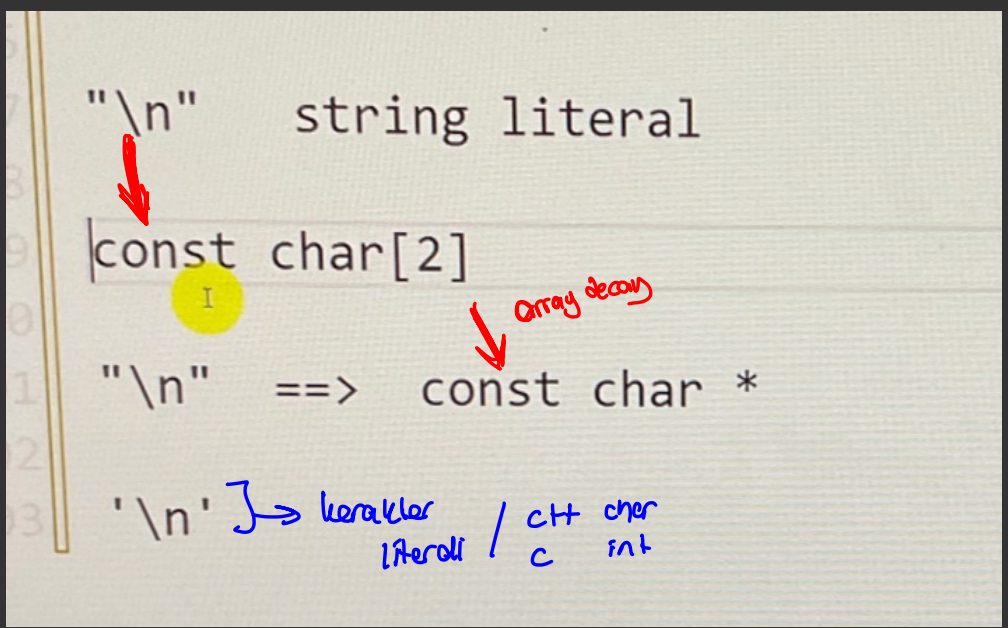
```
string str{ "eray goksu" };

for (auto c : str) {
    cout << c << "\n";
}
```

• Okuma amaçlı kullanımda olur. Fakat değiştirme işleminde yerel değişken değişmiş olur.

→ Değiştirme için auto & c olmalı

→ STL kütüphanesinden devam edicem. Önemli kısımları alt videoda)



```cpp
#include <string>
#include <iostream>
#include <iterator>
#include <vector>

using namespace std;

int main()
{
    std::string s{ "SULEYMAN" };

    s.erase(s.begin() + 3);
    s.erase(3, 1);



}
```

→ Iterator interface

→ index interface

```cpp
std::string s{ "SULEYMAN" };

s.clear();
s.erase();
s.erase(0, std::string::npos);
s.erase(0, s.size());
s.erase(s.begin(), s.end());
s.resize(0);
s = "";
s = {};
s = string{};
s.assign("");
//
```

→ String silmenin 10 yolu

→ Erase valla soruyolarmış