

→ size = capacity olup, insertion  
tolkt gelirse, reallocate yapılır.

→ Avantajı: contiguous/ordinal yarlıdır.  
↳ cache hit/miss orası orta  
↳ indexler erişim, 1. degerin yer  
bilinçli form constant time, Tabii  
değer ne erişim linear complexity.

\* Reallocation takes time !!

Reallocation invalidates pointer and  
Iterators !! → dangling hole gelir.  
→ exception throw etmeden, hataHZIZ oluruz

\* Tek vector kullanırsak, orında pointersının gösterdiği yerler aynı olur. → std::swap  
vector::swap

\* Vektorler default initialize edilirse, size=0, elementi olmayan bir vector olur.

```
//vector<int> ivec;
vector<int> ivec{}; bu servide value ini yapsak da oyn!
std::cout << "ivec.size() = " << ivec.size() << "\n";
std::cout << ivec.empty() << "\n";
```

\*

```
using namespace std;
//vector<int> ivec;
//vector<int> ivec{};
vector<int> ivec(); hileti soru: Bu vec adında vector<int> donduren  
bir fonksiyondur!!!
/*std::cout << "ivec.size() = " << ivec.size() << "\n";
std::cout << ivec.empty() << "\n";*/
```

```
int main()
{
    using namespace std;
    vector<int> ivec(20); 20 adet sağır olan ve hepsi 0'a eft
```

→ bool: false

→ pointer: nullptr.

Fakat, class tarafından specializationda.

```
using namespace std;
vector<Date> dvec(20); bu sınıfın default constructor'u yok
```

```
struct Nec {
    Nec(int);
};

int main()
{
    using namespace std;
    vector<Nec> nvec(20); bu syntax hatası
```

```

int main()
{
    using namespace std;
    vector<int> x(5);
    vector<int> y{ 5 };

```

→ size + parameterized ctor  
 ↑  
 Fehler: Unmöglich!!  
 ↓  
 → initializer list ctor

### \* Fill Constructor:

```

int main()
{
    using namespace std;
    int ival{ 3 };
    vector<int> v1(10, 7); → 10 addt 7
    vector<int> v2(20, ival); → 20 addt ival
    vector<string> v3(10, "bilal"); → 10 addt bilal

    print(v1);
    print(v2);
    print(v3);
}

```

### \* Range für Fülltor Vector / containerlri = file otura → Range Konstrukte.

```

int main()
{
    using namespace std;

    vector<int> ivec{ 1, 5, 7, 9, 12, 76 };
    list<int> ilist{ 1, 5, 9, 2, 0, 3 };

    vector<double> dvec{ ivec.begin(), ivec.end() }; ✓ Busekilde double
    vector<int> v1{ ivec.begin(), ivec.end() }; ✓ Busekilde vector
    cout << v1;
}

```

✓ Busekilde double  
 ✓ Busekilde vector  
 containerlri.

✓ dvec = ivec hat alle

### \* Vector Accessor Methods:

- size() → capacity - size = realloc yapanlar
- capacity() yellow background yer
- empty() → bos mu dogru mu sagusu

## - Vector دالة و انتقام:

```
int main()
{
    vector<string> svec{ "ali", "nur", "eda" , "emre", "yesim", "eray", "volkan" };

    for (size_t i{}; i < svec.size(); ++i) {
        svec[i] → böyle matlak da editir.
    }
}
```

```
int main()
{
    vector<string> svec{ "ali", "nur", "eda" , "emre", "yesim", "eray", "volkan" };

    std::cout << svec[0] << "\n";
    std::cout << svec.front() << "\n";
    std::cout << svec[svec.size() - 1] << "\n";
    std::cout << svec.back() << "\n";
}
```

- Aym [ ] gibt, brachte  
front als referenz abnehmen

- Mutator Fonksiyonları:  $\Rightarrow$  eklenen algirtime yeri:

\* lesRe

→ eger vector, koordinaten beginn bar dopte rechte ecke, sonne O eeler (voluminit)  
 → " " " " " Koek " " " " " , összes elmin

```
vector<Date> dvec;
dvec.emplace_back(12, 12, 2022);
dvec.emplace_back(11, 11, 2011);
dvec.emplace_back(5, 5, 2005);

dvec.resize(7); )→ Fehler in line 8 erraten
print(dvec, "\n");
}
```

```
dvec.resize(0);  
dvec.clear();
```

\* Vector'e ekleme yapma islemleri:

\* push-back → amortised constant-time del. sonar etc.

Fluent API ye warr defi  
return defi yah.

\* employee - back → bairische verben erlernen; perfekt-konjugation nle  
front → concreteness in talk about myself, ourselves, concrete: cognit.

## Insert/emplace:

\* Insert + edition diger en baska organ tekneseti,  $\rightarrow$  begin() " " " en sonde // "  $\rightarrow$  end() ↳ her iki dekomu insert ederse, ise oda

→ Horlama: Front bir **stealer** deg **X**  
Front bir **negate** **V**

\* Insert'in return degeri = Hesicolar.  $\longrightarrow$  Insert edilenin ilkgelen konumu

```
int main()
{
    vector<string> svec{ "eray", "ferhat", "hasan", "emirhan" , "musa", "murat" };
    print(svec);

    auto iter = svec.insert(svec.begin(), "volkan"); ↳ insert edilenin konumu don controller.
    cout << *iter << "\n";
}
```

```
int main()
{
    vector<string> svec{ "eray", "ferhat", "hasan", "emirhan" , "musa", "murat" };
    print(svec);

    bo iter, sezonig'i tutuyor.
    svec.insert(svec.begin(), { "sezai", "recai", "fedai" }); ↳ initialize list constructor.
    print(svec);
}
```

→ eger ogeleri bilgirebil, tarihe bol loop ile push back yapmakten daha az maliyetli

```
svec.insert(svec.end(), 10, "recep"); single element fill komutu.
print(svec);
}
```

→ benzeri手段, range olarak ne, bir vektore, bir liste eklenebilir. (eski notlarda var)

\* Atama islemleri: ( $v_1 = v_2$ )

```
int main()
{
    vector<string> v1{ "eray", "ferhat", "nuri", "selim", "turgut"};
    vector<string> v2{ "aliye", "nedime", "saliha", "sumeyye", "zarife"};

    v1 = move(v2); ↳ move assignment
    v2 = v2; ↳ copy assignment
}
```

\* Fakat **istikallik containerlar** bittiğinde  
atamamız X

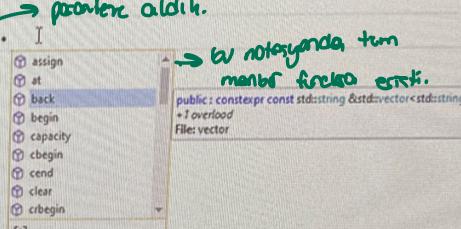
\* Bunu tam assign() kullanımlı

```
int main()
{
    vector<string> svec{ "eray", "ferhat", "nuri", "selim", "turgut"};
    list<string> slist{ "aliye", "nedime", "saliha"};

    print(svec);
    svec.assign(slist.begin(), slist.end()); ↳ ist beginin ist end'e vektore standi.
    print(svec);
}
```

↳ Bir diger notasyon:

```
(svec = { "ece", "eda", "nun", "gul" }). ↳ paranteze olduk.
print(svec); Inistruksiyon ist
```



```
svec.assign(20, "tayyip");
print(svec);
```

*assign first ile hemen hemen bir*

\* her eleme fonksiyonu karsılık bir emplace function verdir! |

```
vector<string> svec{ "eray", "ferhat", "nuri", "selim", "turgut"};
print(svec);

svec.insert(svec.begin(), "ali"); i
svec.emplace(svec.begin()); [ ] default int eden vector bosca empty string var.

print(svec);
```

```
int main()
{
    vector<string> svec{ "eray", "ferhat", "nuri", "selim", "turgut"};
    print(svec);

    svec.emplace(svec.begin(), 5, 'A'); AAAAAA silindikte kalkar ciler.
    print(svec); (1 elemen ekledik osimde)
String sınıfının ctor'una forward edildi.
```

eray ferhat nuri selim turgut

*AAAAA eray ferhat nuri selim turgut*

## \* Silme Fonksiyonları:

- Konuma göre silme
- Lange orijini silme

```
int main()
{
    vector<string> svec{ "eray", "ferhat", "nuri", "selim", "turgut", "volkan",
    "naci" };

    while (!svec.empty()) {
        print(svec);
        svec.pop_back(); son üyesi siler. / Return değer yok!
        (void)getchar();
    }
}
```

```
int main()
{
    vector<string> svec{ "eray", "ferhat", "nuri", "selim", "turgut", "volkan", "naci" };

    string name;
    std::cout << "silinecek ismi girin: ";
    cin >> name;

    if (auto iter = find(svec.begin(), svec.end(), name); iter != svec.end()) {
        svec.erase(iter);
        print(svec);
    }
}
```

*global find konteynerni kullandi*

*if with initializer*

*name'i arattı.*

# eger burada ferhat silirse, erase'in döndüreceği iter → nuri

- Insertler, insert edilmiş ilk konumu döndürür.
- Eroşeler, erase edilmiş ögekonunun konumu döndürür.

```
//svec.clear();  
//svec.resize(0);  
//svec.erase(svec.begin(), svec.end());  
//svec = vector<string>{};  
//svec = {};  
svec.assign({});  
svec.assign(vector<string>{})
```

\* Tüm elementler silme yarantır. (İmdekkotta şurda)

\* Silme işlemi size`ı artırır, capacity`ı artırır. Capacity artırılmadan, shrink to fit! kullanımlı.

folket by non-binding

shrink request. (Nero dengesizliğinden

shrink ileşti).