* Inline Functions: → inline expansiondan farkını bil.

→ Inline Expansion:
- Bir optimizasyon tekniği (compiler yapar)
- Fonksiyonun çağırıldığı yerin geçici çıktısını tutup, linker'a iletmek yerine direkt çağırıldığı yerde execute edilir.

- Bir fonksiyon inline olarak tanımlanıp, header'a yazılırsa, ODR kuralının ihlal edilmediğini garantiler.

- Eğer headerdaki tüm fonksiyonlar, inline tanımlanırsa, bu durumda cpp dosyası olmadan modül oluşturulabilir. Böyle modüllerden oluşan kütüphanelere, header-only library denir.

- Inline fonksiyon tanıtarak, compiler'a inline expansion ŞANSI verilir. (Garantilemez)

- C++ 17 ile inline variables eklendi.
  → tanıtıldığı header'ı ekleyen tüm dosyalarda sadece 1 adet tanım olduğunu garantiler.

- Global fonksiyonlarda olduğu gibi, member functionlar da inline olarak header'da tanıtılabilir. Bu durum ODR'ı ihlal etmez.

```
class Murat {

public:
    void set(int x);
private:
    int mx;
};


inline void Murat::set(int x)
{
    mx = x;
}
```
→ ODR violation yok.

- Direkt Class Definition içerisinde de fonksiyon tanıtılabilir. Implicitly Inline olur.

```
class Murat {

public:
    void set(int x)
    {
        mx = x;
    }
private:
    int mx;
};
```
→ Set implicitly inline fonksiyon oldu

* Constructors: - Sınıf nesnesini hayata getiren üye fonksiyon / Ömrünü sonlandıran da destructor

- Bir sınıfın constructor ve destructorları Global veya Static olamaz!!
- Constructorın adı class adı ile Aynı
- Constructorların return kavramı yoktur.
- Constructorlar Overload edilebilir.

* Bir nesnenin, argüman almadan çağrılabilen constructor'ı → Default ctor
→ Ya argüman almayacak, ya da default argument alacak.

→ Special Member Functions: - Default Constructor
- Destructor
- Copy Constructor
- Move Constructor
- Copy Assignment
- Move Assignment

⇒ Derleyici biz yazmasak bile, bu fonksiyonları kendi generate edebilir. / Buna default da denir.

* Constructor ve Destructor CONST OLAMAZ.
- Constructor ve Destructor PRIVATE OLABİLİR ama Client ulaşamaz. Protected olabilir
- Sınıfın içinde inline tanımlanabilir.
- Constructor, non-static olduğu için, this pointer kullanılabilir.

* Destructors:

- Constructor gibi ama başında ~ (tilda) var.
- Global olamaz, static olamaz, const olamaz.
- Parametresi OLAMAZ!! Overload edilemez!

* Statik Ömürlü Nesnelerde Constructor / Destructor. Statik ömür → ya global olarak tanımlanacak.
→ ya da static anahtarıyla

1. Global olarak tanımlanır:

```
};                                  // Bu bir class'ın hem ctor hem de dtor'u var

Nec nx;
Erg ex;                             // main fonksiyonundan önce construct edilir
                                    //  "      "     bitmeden hemen önce destruct edilir.
int main()
{
    std::cout << "main basladi\n";
    std::cout << "main sona eriyor\n";
}
```

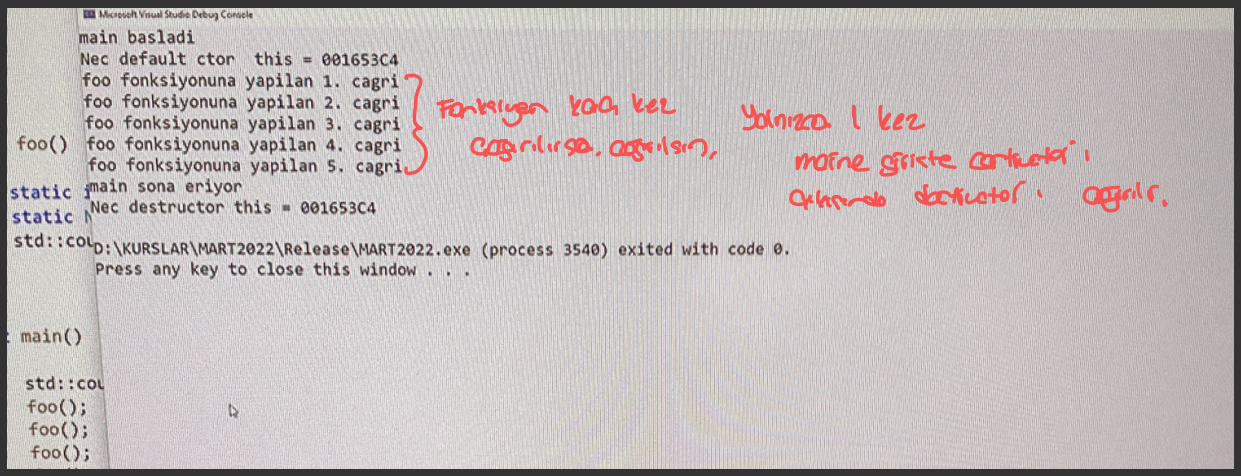
Fakat
1. Nec ctor
2. Erg ctor
3. Erg dtor
4. Nec dtor

→ en son construct edilen, ilk destruct edilir.

* Fakat, farklı dosyalarda hangi constructor önce çağrılır belli değil.
↓
Static Initialization fiasco

2. Statik Ön evrim → Bir fonksiyonun statik yerel değişkeni

- Bu nesnenin construct edilmesi için tanımlı olduğu fonksiyonun çağırılması gerekir.
- Fakat statik ömürlü olduğu için



Fonksiyon kaç kez çağırılırsa, çağrılsın, yalnızca 1 kez nesne girişte constructor, otomatik destructor çağırılır.

* Otomatik Ömürlü Sınıf Nesneleri: → Bir fonksiyon içinde tanımlanan nesneler otomatik ömürlüdür.

- Fonksiyon kaç kez çağırılırsa, bu sınıf nesneleri o kadar construct ve destruct edilir.
- Array elemanı class olabilir.
  - ↳ tüm array elemanları için tek tek constructor / destructor çağırılır.