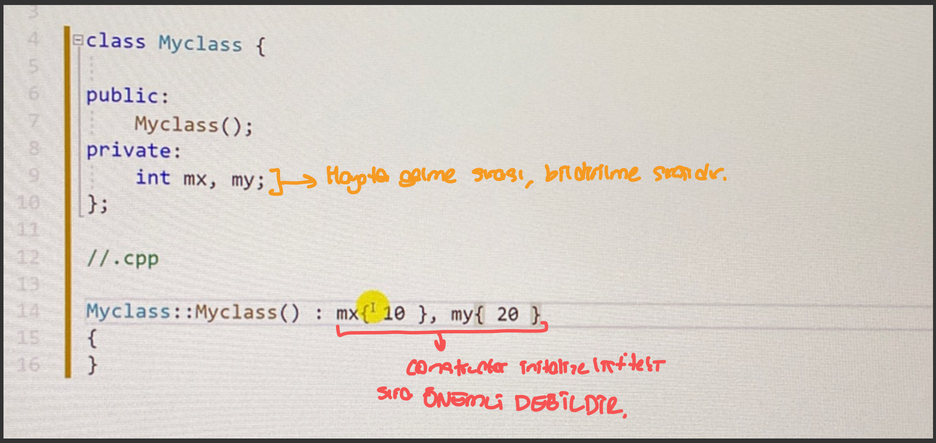
*** Tekrar:** - Eğer tanıtılmadıysa, compiler default constructor default eder. Fakat biz default olmayan bir constructor yazarsak Compiler Default etmez.

*** Constructor Initializer List:**



```cpp
class Myclass {

public:
    Myclass();
private:
    int mx, my;
};

//.cpp

Myclass::Myclass() : mx{ 10 }, my{ 20 }
{
}
```

mx, my → Hayata gelme sırası, bildirilme sırasıdır.

Constructor initializer list sıra ÖNEMLİ DEĞİLDİR.

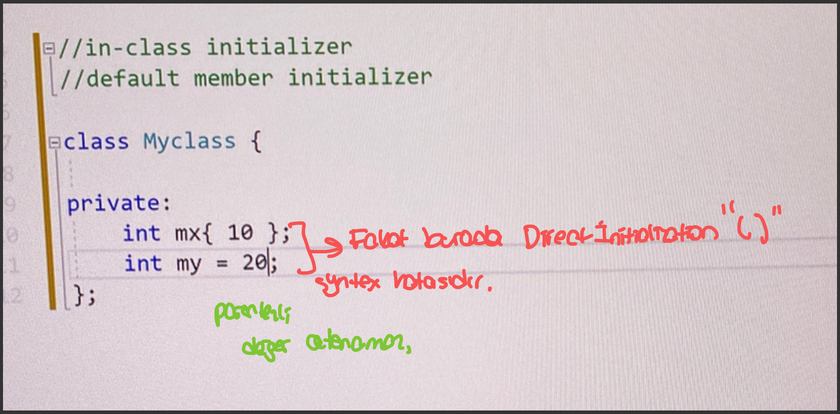
* Class constructor'a gelmeden, eğer değer atanmadıysa, default olarak init edilir. (çöp değer)

* Ancak CONST NESNELER Default init yapılamaz.

* Her zaman ilk tercih bu olmalı. (Eğer mümkünse) Çünkü efficiency → Default initleyip atama yapmaktan hızlı.

* Sınıfın data member'ı const ya da referans ise, constructor initializer list ZORUNLU

*** In-class / Default Member Initialization:**



```cpp
//in-class initializer
//default member initializer

class Myclass {

private:
    int mx{ 10 };
    int my = 20;
};
```

→ Fakat burada Direct-Initialization "()" syntax hatasıdır.

parantezli değer atanamaz.

⇒ Fakat Burada fiilen ilk değer vermiyoruz.

⇒ Bununla compiler'a örtülü olarak, constructor init. list'e ekletiyoruz. Eğer constructorda değer olmazsa.

⇒ Eğer hem default member init, hem constructor init list varsa, Constructor init list dikkate alınır.

*** Delegating Constructor:**

→ Bir sınıfın constructor'ı data member initlemek için başka constructorları çağırabilir. Buna delegating constructor denir.

→ Delegating Constructor varsa başka constructor init list kullanılamaz!!

* Special Member Functions:

- default ctor
- destructor
- copy ctor
- move ctro
- copy assignment
- move assignment

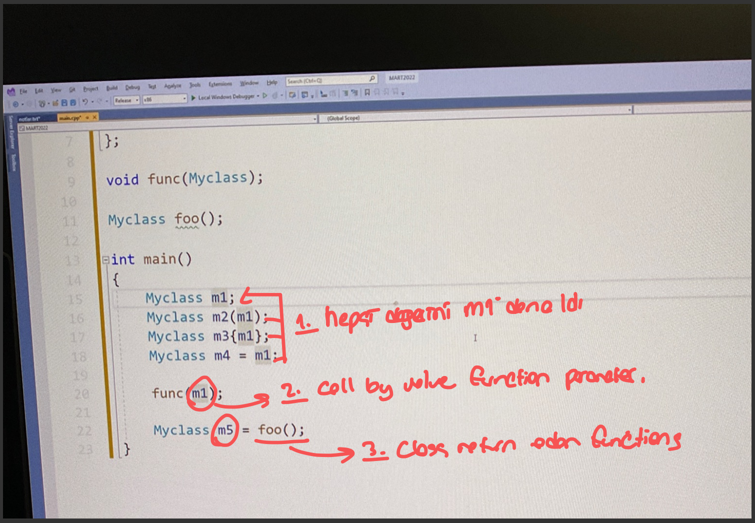
copy ctor, copy assignment → copy members
move ctro, move assignment → move members

→ Özel koşullarda compiler bunları kendi default eder. Biz bildirmesek de.

→ Eğer derleyici sınıfın bir özel üye fonksiyonunu default edecekse fakat derleyicinin özel üye fonksiyonu oluşturma sürecinde syntax hatası oluşursa derleyici özel üye fonksiyonu delete eder. Priv erişim, const, ref erişimi gibi... !!

– Copy Constructor:

- Kopyala. değerini aynı türden bir başka sınıf nesnesinden olarak geliyor.
- Kullanıldığı senaryolar:

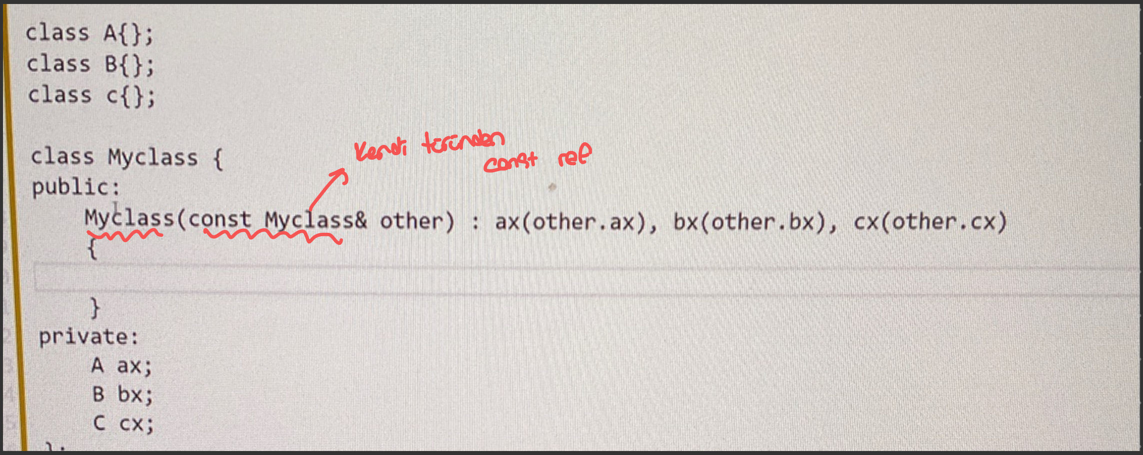


```
};

void func(Myclass);

Myclass foo();

int main()
{
    Myclass m1;
    Myclass m2(m1);
    Myclass m3{m1};
    Myclass m4 = m1;

    func(m1);

    Myclass m5 = foo();
}
```

1. hepsi değerini m1'den alıyor
2. call by value function parameter.
3. Class return eden functions

- Rule of Zero: Bütün special member fonksiyonların, compiler tarafından default edilmesi

- Non-static, inline, public olarak default eder compiler.



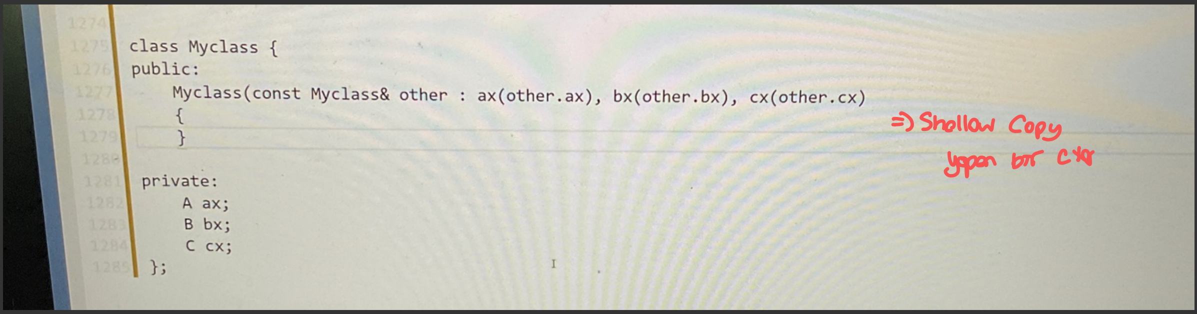
```
class A{};
class B{};
class c{};

class Myclass {
public:
    Myclass(const Myclass& other) : ax(other.ax), bx(other.bx), cx(other.cx)
    {

    }
private:
    A ax;
    B bx;
    C cx;
};
```

- Sınıfın veri elemanlarından biri Eğer POINTERSA / Referanssa → COPY CTOR, o pointerın değerini kopyalar.
  ↳ Yani aynı nesneye işaret eder - Bu dangling pointera sebebiyet verebilir.

- Shallow Copy: Kaynağı değil, kaynağı gösteren pointer'ın kopyalanması
- Deep Copy: Kaynağın kendisini kopyalar.

- <u>RAII:</u> Resource Aquistion is Initialization
  - → Otomatik yapı
  - → Bir sınıf nesnesi işlevini yerine getirmek için kaynak edinmeli

→ tekrar izle!!! son 15 dk.



```cpp
class Myclass {
public:
    Myclass(const Myclass& other : ax(other.ax), bx(other.bx), cx(other.cx)
    {
    }

private:
    A ax;
    B bx;
    C cx;
};
```

⇒ Shallow Copy yapan bir c-tor