Ders 44

```
definition

================================================================================

C'de her declaration bir definition olmak zorunda değil
    non-defining declaration


ancak her definition aynı zamanda bir declaration
```

```
void func(int);
```

Bu decralation ama difiniton değil her bilgi yon

Non defined definition

```
int x = 5;
```

Bu hem declarion hemde definition

	object name	function name	label name
scope	*yes*	*yes*	*yes*
type	*yes*	*yes*	*no*
storage duration	*yes*	*no*	*no*
linkage	*yes*	*yes*	*no*

Örnek fonksiyon yazı ileticekse

```
// çağırandan dizi adresi iste yazıyı o diziye yaz
// statik yerel bir dizi oluştur. yazıyı oraya yaz. dizinin adresini döndür

char* get_random_psw(size_t len)
{
    static char buffer[200];

    for (size_t i = 0; i < len; ++i)
        buffer[i] = rand() % 26 + 'A';

    buffer[len] = '\0';

    return buffer;
}
```

Ama her çağrıda son değeri yazdırıcak bir önceki değerkaybolucak.

Farklı bir ctime static nesne adresi döndürüyor.

```
int main(void)
{
    time_t timer;
    time(&timer);

    for (int i = 0; i < 10; ++i) {
        printf("%p\n", ctime(&timer));
    }


}
```

Örnek 2. Tema

```c
#define URAND_MAX           20

int urand(void)
{
    static int flags[URAND_MAX] = { 0 };
    static int count = 0;

    if (count == URAND_MAX)
        return -1;

    int ival;

    while (flags[ival = rand() % URAND_MAX])
        ;

    flags[ival] = 1;
    ++count;
    return ival;
}
```

Static olmasa eşsiz sayı üretemiyecekti

Nesneler global olsada olurdu ama sadice bu fonksiyon kullansın istediğim için böyle yaptım.

Örnek idiom ilk kez çağırılınca yapılacak işler

```c
void func(void)
{
    static int first_call_flag = 1;

    if (first_call_flag) {
        printf("ilk cagrida yapilan isler\n");
        first_call_flag = 0;
    }

    printf("her cagrida yapilan islemler\n ");
}


int main(void)
{
    for (int i = 0; i < 10; ++i) {
        func();
    }

}
```

İlk kez çağırılınca yapılacak işler

```
void func(void)
{
    const int primes[] = {
      2,   3,   5,   7,  11,  13,  17,  19,  23,  29,
     31,  37,  41,  43,  47,  53,  59,  61,  67,  71,
     73,  79,  83,  89,  97, 101, 103, 107, 109, 113,
    127, 131, 137, 139, 149, 151, 157, 163, 167, 173,
    179, 181, 191, 193, 197, 199, 211, 223, 227, 229,
    233, 239, 241, 251, 257, 263, 269, 271, 277, 281,
    283, 293, 307, 311, 313, 317, 331, 337, 347, 349,
    353, 359, 367, 373, 379, 383, 389, 397, 401, 409,
    419, 421, 431, 433, 439, 443, 449, 457, 461, 463,
    467, 479, 487, 491, 499, 503, 509, 521, 523, 541,
    };

    //

}
```

Burada dizi otomatik olduğu için her seferinde fonksiyon içinde bu diziyi belleğe yazıcaz ve silicez 1000 kez yazılırsa sıkıntı olur ama static yapsaydık bu şekilde her seferlnde belleğe yazmaya gerek kalmıcaktı.

```
void func(void)
{
    static const int primes[] = {
      2,   3,   5,   7,  11,  13,  17,  19,  23,  29,
     31,  37,  41,  43,  47,  53,  59,  61,  67,  71,
     73,  79,  83,  89,  97, 101, 103, 107, 109, 113,
    127, 131, 137, 139, 149, 151, 157, 163, 167, 173,
    179, 181, 191, 193, 197, 199, 211, 223, 227, 229,
    233, 239, 241, 251, 257, 263, 269, 271, 277, 281,
    283, 293, 307, 311, 313, 317, 331, 337, 347, 349,
    353, 359, 367, 373, 379, 383, 389, 397, 401, 409,
    419, 421, 431, 433, 439, 443, 449, 457, 461, 463,
    467, 479, 487, 491, 499, 503, 509, 521, 523, 541,
    };

    //

}
```

Bunların zaten değişmeyeceğini bildiğimiz için static yapın daha verimli olur.

//

Soru değişken x hangisidir. X ifarklı dosyadan kullanma

```
================================
ahmet.c

int x = 10;



mehmet.c
========

void func(void)
{
    x = 99;
}
```

Burada kullanılan x ahmetteki ama derleyici bunu nameloop uptabulamaz nasıl olucak

```
ahmet.c

int x = 10;



mehmet.c
========

extern int x;

void func(void)
{
    x = 99;
}
```

Extern int x ile kodun başka dosyada olduğunu ordan almasını söyledik.

Burada kullanılan x değişkeni Ahmet .c de tanımlanan x değişkeni eğerki Mehmet.c içinde x i başka yerde tanımlamaışsam bunu kullanır ve ahmetteki x i 99 yapar.

```c
//
extern int x;

int main(void)
{
    x = 56;

}
```

Burada derleyici extern int bildirimi için bellekte yer ayırmayacak bunu başka dosyadan kullanması gerektiğini biliyor.

Peki bu x in bildiriminin nerde olması gerekir

Cevap başlık header dosyası.

```
=========================================
necati.c                    necati.h

int x = 10;                 extern int x;
```

Diğer kaynak dosyalar necati .h yı include ederek x i kullanması gerektiğini bilecek.

```
=========================================
necati.c                    necati.h

int x = 10;                 extern int x;
static int g = 10;          -------------
```

Sadece kendi dosyamda kullanacaksam static yazıp başlık dosyasına bildirim koymayacağız.

```
=========================================
necati.c                    necati.h

int x = 10;                 extern int x;
static int g = 10;          -------------

void func(int x)            void func(int);
{
    ///...
}
```

Fonksiyonlar içinde bu geçerli bildirime extern koymasakta default olarak zaten extern olduğu için koymasakta olur.

En güzel örnek nutility. Dosyalarımız.

```
#include <stdio.h>


static int icmp(const void* vp1, const void* vp2)
{
    return *(const int*)vp1 - *(const int*)vp2;
}
```

Nutility de böyle bir tanımlama yaptık bunu qsort kullansın diye oluşturdum bu yüzden dışarıdan kullanılmasın diye de static ön eki kullandım.

```
int icmp(const void* vp1, const void* vp2)
{
    return *(const int*)vp1 - *(const int*)vp2;
}
```

Static olarak tanımlamasam ve başlık dosyasına bildirim koymasamda yanlış olur.

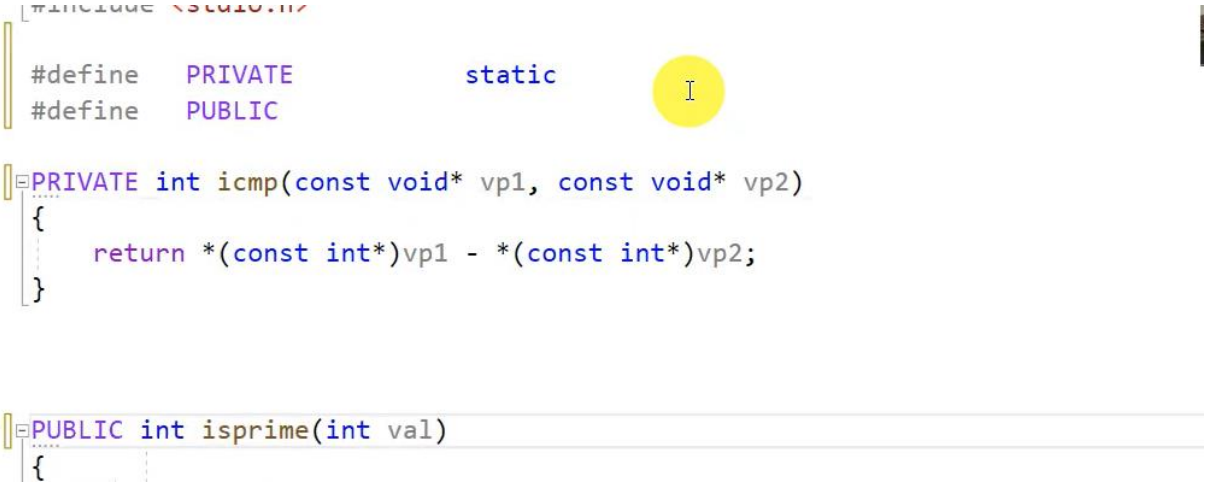
Bu sefer diğer dosyalardaaynı isimli fonksiyonu dış bağlantı yapmak istesem bu sefer link hatası olacak.

```
#define   PRIVATE          static

PRIVATE int icmp(const void* vp1, const void* vp2)
{
    return *(const int*)vp1 - *(const int*)vp2;
}
```

Bazı programcılarbunu pravite yazarak yapıyolar ama bunuda bir makro kullanarak yapıyolar bunu yapma nedenleri nesne yönemlimi programlamaya benzetmek.

Bir hile daha var.



```
#include <stdio.h>

#define   PRIVATE           static
#define   PUBLIC

PRIVATE int icmp(const void* vp1, const void* vp2)
{
    return *(const int*)vp1 - *(const int*)vp2;
}
```



```
PUBLIC int isprime(int val)
{
```

Burda public yazıyolar dışa açık gibi ama bu macroda hileli çünkü makroyu yazıp yanına bişey yazmazsak ön işlemci bunu siliyor ve derleyici bu yazıyı görmüyor yanı sadece kodu yazan görüyor.