Ders 18

Örnek makro fonksiyon farkı

```
#define square(x)        ((x) * (x))

int foo(void);

int main()
{
    int y = 10;

    int result = square(foo());

}
```

Makro ile çağırırken 2 kez çağırılış olacak burda fark olabilir.

Bir değeri değiştiriyor olabilir.

Örnek, önce hangisi çalışır

```
#include <ctype.h>

int square(int a)
{
    printf("square islevi cagrildi!\n");
    return a * a;
}

 #define square(x)        ((x) * (x))


int main()
{
    int ival = 5;

    int result = square(ival);

}
```

Önce makro çalışır ön işlemci program derleyiciden önce square yerine makroyu yazdı.

2.

Fonksiyon kullanmak için ne yapmak gerekirdi.

```c
int square(int a)
{
    printf("square islevi cagrildi!\n");
    return a * a;
}

#define square(x)        ((x) * (x))


int main()
{
    int ival = 5;

    int result = (square)(ival);

}
```

Squareyi paranteze almak fonksiyonu çağırır. Makronun çalışması için isimden sonra açılan parantez gelmesi gerekir.

3. neden sentak hatası var

```c
int square(int a)
{
    printf("square islevi cagrildi!\n");
    return a * a;
}

#define square(x)        ((x) * (x))


int main()
{
    int ival = 5;

    int result = square(ival);

}
```

Ön işlemci programı fonksiyonunda adını değiştir.

Birlikte tanımlanacak ise define fonksiyonun altına yazılmalı ön işlemci yazıldığı yerin altına işlem yapar.

Yada fonksiyonu parantez içinde yazarsak makro fonksiyonun adını değiştirmez (önemli tanımlamayı böyle yap.) (func)(void)

Örnek iç içe makro

```
#define isupper(c)     ((c) >= 'A' && (c) <= 'Z')
#define islower(c)     ((c) >= 'a' && (c) <= 'z')

#define isalpha(c)  (isupper(c) || islower(c))

int main()
{
    int ch = 'D';

    (((ch) >= 'A' && (ch) <= 'Z') || ((ch) >= 'A' && (ch) <= 'Z'))

}
```

Son terim yerine isalpha(ch) yazarsak ön işlemci önce onu sonra islower ve isupperı açacak

```
#define _CRT_SECURE_NO_WARNINGS

#define     asize(x)        (sizeof(x) / sizeof(x[0]))
#define     ran_elem(a)     a[rand() % asize(a)]

int main()
{
    int ar[] = { 2, 7, 9, 1, 5, 10, 34 };

    ar[rand() % (sizeof(ar) / sizeof(ar[0]))]


}
```

Son terimde ran_elem(ar) yazıyor.

Örnek makro çalışınca ne olur.

```
#define printf(x)     printf("%d\n", x)


int main()
{
    int ival = 10;

    printf(ival);

}
```

Printf açılınca bir daha print yazılıp açılması lazım ancak makro içinde aynı isim varsa makro sadece bir kez açılır.

Örnek int türü makro

```
int main()
{
    int x = 10;
    int y = 20;
    int z = 30;

    printf("%d\n", x);
    printf("%d\n", x + y);
    printf("%d\n", x * x + y * y + z * z);

}
```

İfadesini makro ile yaoalım.

```c
#define  iprint(a)      printf(#a "= %d\n", a)

int main()
{
    int x = 10;
    int y = 20;
    int z = 30;

    iprint(x);  //x = 10
    iprint(x + y); //x + y = 30
    iprint(x* x + y * y + z * z); //x * x + y * y + z * z =

}
```

#a ifadesi print içinde string makroyu yazar.

Örnek ## kullanımı

```c
#define uni(x, y)       x##y

int main(void)
{
    int prime_count = 0;

    ++uni(prime_, count);

    printf("prime_count = %d\n", prime_count);
```

X ve ifadesini birleştirdi.

Örnek iyi

```c
#include <stdio.h>

#define make_swap_fn(t)  void swap(t *p1, t *p2) { \
                         t temp = *p1; \
                         *p1 = *p2; \
                         *p2 = temp;}


make_swap_fn(int)
make_swap_fn(char)
make_swap_fn(double)
make_swap_fn(long)

int main()
{
```

Burada benzer işleri yapıcak ama türleri değişik olan fonksiyonları nasıl makro ile yapabileceğimizi görüyoruz. Makroru ön işlemci programa fonksiyon yazdırmak için kullandık sonuç olarak makrolar kullanıldığı yerlerde sağ tarafındaki yazıyı olduğu gibi kodayazar ve derleyiciye iletir. Ancak bir sorun var aynı isimle birden fazla fonksiyon tanımlamış oluyoruz bunda cde geçerli değil sorunu ## ön işlemci operandı ile aşacağız

```c
#include <stdio.h>

#define make_swap_fn(t)  void swap_##t(t *p1, t *p2) { \
                         t temp = *p1; \
                         *p1 = *p2; \
                         *p2 = temp;}


make_swap_fn(int)
make_swap_fn(char)
make_swap_fn(double)
make_swap_fn(long)

int main()
{
```

Artık makro açıldığında isimi swap değl

Swap_int

Swap_char

Swap_double olacak.

Örnek mülakatlarda çıkar.

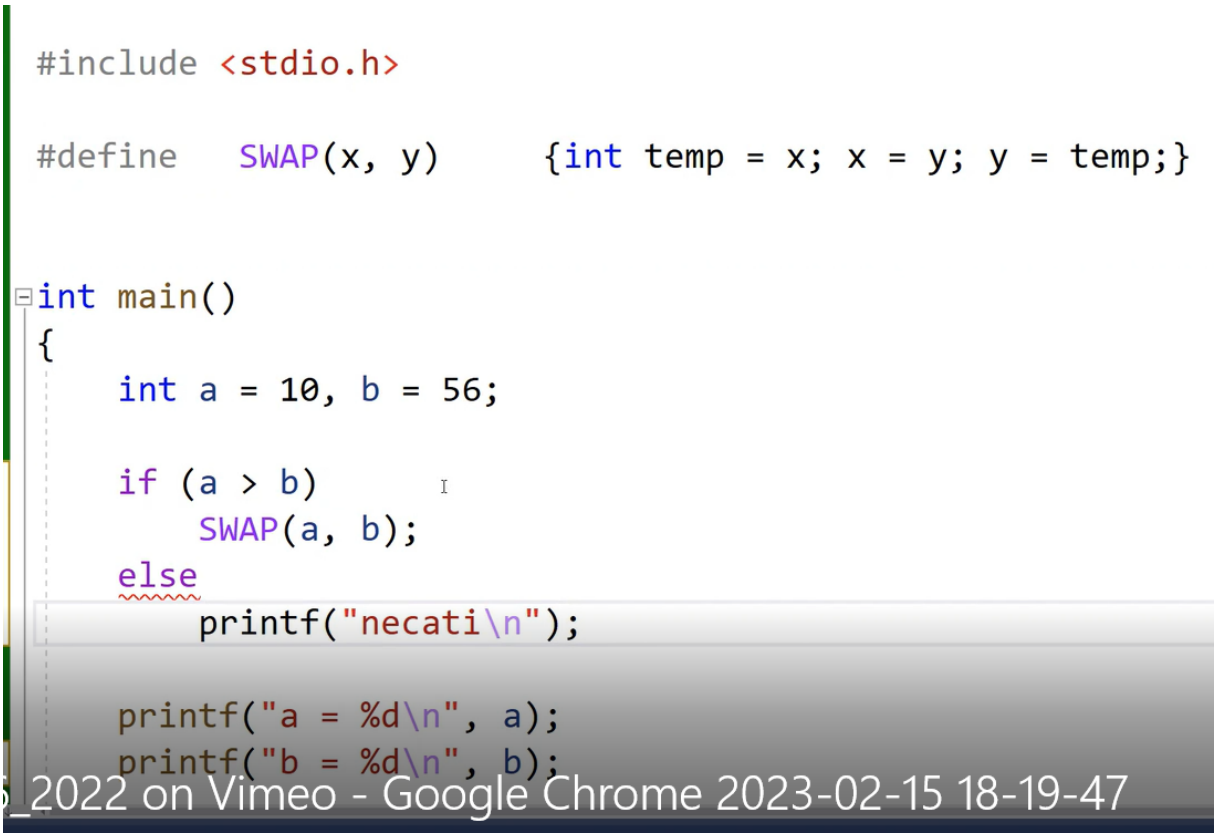
Birden fazla expression içeren makro yazalım.

```c
#include <stdio.h>

#define   SWAP(x, y)     {int temp = x; x = y; y = temp;}

int main()
{
    int a = 10, b = 56;

    SWAP(a, b);

    printf("a = %d\n", a);
    printf("b = %d\n", b);
```

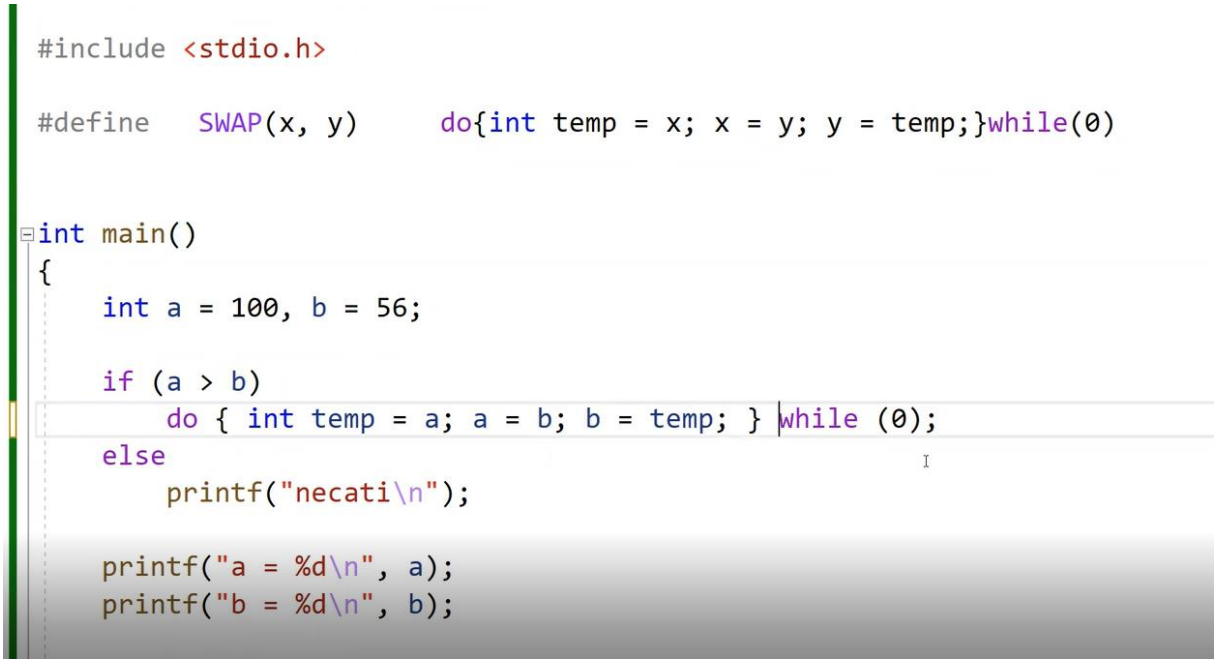
Bu kodda sıkıntı yok ancak kod açılınca ektra sonuna noktalı virgül gelecek.

```c
#include <stdio.h>

#define   SWAP(x, y)     {int temp = x; x = y; y = temp;}

int main()
{
    int a = 10, b = 56;

    {int temp = x; x = y; y = temp; };

    printf("a = %d\n", a);
    printf("b = %d\n", b);
```

Ektra null statement var örnek if ile olsaydı.



Bu örnekte 2. Bir ; virgül olması else ifadesini boşa çıkaracak ve kodda hata yapacak



Do while eklediğimizde sentax doğru while den sonra noltalı virgül gelecek. 0 olduğu için bir kez çaışacak.

Örnek # if #endif