Ders 45

```
int main()
{
    const int x = 10;
    int* p = (int*)&x;

    *p = 99;

}
```

Burada kod legal olabilir ancak tanımsız davranıştır.

Değeri değişmeyecek sabit gibi kullandım ve const kullanmadım bunun zararı nedir

```
int main()
{
    int size = 100;

}
```

Okuyanı yanıltır. Kodda yanlışlıkla değiştirmeye karşı korumaz. Optimizasyonu etkiler ve değiştirince hata vermez. En önemlikalite göstergesi

Dizilerde const değişken olabilir mi

```
int main()
{
    const int x = 10;

    //int a[x];

```

Bu switch yada dizi boyutunda kullanmam. Hata

Derleyici burada hata veya uyarı verir mi

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

İki bidirimde aynı oluyor ve bildirimde const olması bişey değiştirmiyor hata olmaz bildirim olduğu için hata yok.

Hata varmı

```
int main()
{
    int x = 10;
    int y = 67;

    const int* ptr = &x;

    //ptr = &y;
}
```

Hayır burada değişmeyecek olan *ptr değeri ptr adresi değişebilir.

```
int main()
{
    int x = 56;
    int y = 123;

    int* const p = &x;

    p = &y;


}
```

Burada sentax hatası ama *p atamasında sıkıntı yok

Bu hatalı mı

```
int main()
{
    int x = 56;

    const int* p = &x;

    //x = 98;
```

Hayır burda sadece pointer nesneyi değiştirmesin demek ama nesne kendini değiştirebilir.

Hangisi sentax hatası

```
typedef int* iptr;

int main()
{
    int x = 10;
    int y = 20;
    const iptr p = &x;

    //p = &y;
    //*p = 56;
```

Özel const özelliği burada p = &y ataması hatalıdır.

Hatalı mı

```
void func(int* p);
//void func(int *const p);


int main()
{
```

2 bildirim olması promlem değil top level const parametre constluğu bir hata yaratmaz.

Tanımsız davranış var mı

```
int main()
{
    int x = 10;
    const int* p = &x;

    int* ptr = (int*)p;

    *ptr = 9999;
}
```

Hayır yok isteyerek int* türüne çevirdim ve x fiziksel const değil sorun yok.

Legal ve doğru mu

```
int main()
{
    char* p = "ertan";
}
```

Legal ancak doğru değil ama string literali değiştirelemez bunu *p ile değiştirebilirim bu yüzden

Const char* p de saklamalıyım.

```
int main()
{
    int x = 10;
    const int* p = &x;
    *p = 13;
```

Bu sentax hatası

```
int main()
{
    int x = 10;
    const int* p = &x;

    *(int *)p = 12;
```

Ama bu hata değil tür dönüştürme yapmışım.

```
const int* foo(void);

int main()
```

Burada adresi döndürülen yerdeki nesneyi değiştiremeyiz

.

Örnek valotine olan hangisi

```
#define  PRXT   (int *)0x1ac40000

int main()
{
    volatile int* p = PRXT;

    *p

}
```

*p nın değeriadrsteki nesne değiştirilebilir demek.

```
#define  PRXT   (int *)0x1ac40000

int main()
{
    int* volatile p = PRXT;
```

Buda p nin kendisi değiştirilebilir demek olacak.

```
#define  PRXT   (int *)0x1ac40000

int main()
{
    volatile int* volatile p = PRXT;

    *p
```

Burada 2 side dışardan değiştirilebilir bunun değerini optimize etme her seferinde bellekten çek demek.

Burada mantık hatası var mı

```
#define  PRXT   (int *)0x1ac40000

int main()
{
    const volatile int* p = PRXT;
```

Ben ilk olarak var dedim const değiştirilemeyen volatile dış mihraplar tarafından değiştirilebilen demek.

Ama burada const değişkeni kodda değiştirmemem gerektiğini volatile da bu nesnenin değerini her zaman bellek alınması gerektiğini anlatıyor dışarıdan müdahale ile değişmesinde bir sakınca yok const için hata kod ile değişmesi.

Structureler

Örnek yapı elemanları

```
struct Date {
    //
};

struct Person{

    int x, y, z;
    double d;

};
```

Yapı elemanları int double pointer fuction pointer hatta başka bir struct türü olabilir ancak

Fonksiyon olamaz.

Peki derleyici bunu görünce bir yer ayırıyor mu

```
struct Data {
    int a, b, c;
};
```

Hayır bu sadece derleyiciye bir bilgi veriyor . bu tür ile değişken tanımlamadan derleyici yer ayırmaz.