

# **C**

## **Pointers**

### **(İşaretçiler)**

Dr. Öğr. Üyesi M. Ozan AKI

# İşaretçiler (Pointer)

İşaretçi (Pointer), bir değişkenin bellekte bulunduğu adresi tutan değişkendir.

C dilinin kendinizi topuğunuzdan vurmanıza olanak sağlayan yegane yapıları işaretçiler (pointer) dır.

C#, Java, gibi yüksek seviyeli dillerde doğrudan ya da kontrolsüz bir şekilde pointer kullanımına izin verilmez. Ancak Pointer kullanabilmeniz için kodu UNSAFE olarak belirtmeniz gerekir.

Pointerlar, birçok algorithmada verilere hızlı ve sistematik bir erişim sağladığından uygulanabilir.

# İşaretçiler (Pointer)

C Dilinde Pointer tanımlamaları için iki işaretçi kullanılır:

**\*** : Veri tipinin hemen sağına ya da değişken adının soluna yazılarak, belirtilen tipteki değişken için işaretçi tanımlar.

**\*** : Tanımlanmış herhangi bir pointer önüne yazıldığı zaman ise, o adresin içeriğini döndürür.

**&** : Tanımlanmış herhangi tipteki değişkenin bellek adresini döndürür. Bu adres ancak aynı tipte tanımlanmış bir pointer değişkene atanabilir.



# İşaretçiler (Pointer)

```
int i;           // i adında int tipinde bir değişken

int *iptr;       // int tipinde bir pointer tanımlanır

i = 5;           // i değişkenine 5 değerini ata

iptr = &i;       // i değişkenin adresini iptr'ye ata

*iptr = 3;       // iptr'nin gösterdiği adresteki değeri 3 yap

printf("%d", i); // i değişkenini ekrana yaz
```



# Pointer Aritmetiği

Pointerlar üzerinde yapılan aritmetik toplama ve çıkarma işlemlerinde birim artış değeri, tanımlandığı veri tipinin bellekte kapladığı alan kadar olmaktadır.

Böylece ardışık bellek alanlarında bulunan aynı tipteki verilere erişim kolaylıkla mümkün hale gelir.

Örneğin,

```
char *cptr;
```

```
cptr++; // cptr değeri 1 birim (1 byte) artar
```

```
int *iptr;
```

```
iptr++; // iptr değeri 4 birim (4 byte) artar
```

# Pointer ve Diziler

İşaretçi ve Diziler çalışma prensipleri oldukça benzerdir. Bu nedenle çoğu zaman herhangi dizi, pointerlar ile kullanmaya çok uygundur.

```
int dizi[10];  
int *ip;
```

```
ip = dizi;
```

```
dizi[3] = 5; // ifadesi ile aşağıdaki ifade aynıdır;  
*(ip+3) = 5; // değeri pointer yoluyla aktarma
```



# Pointer ve Diziler

Ekrandan girilen bir karakter kümesini ekrana tersten yazan C programını;

- 1) Sadece dizileri kullanarak,
- 2) Pointer kullanarak gerçekleştiriniz.

Ekrandan veri girişinde aşağıdaki iki fonksiyonun farkı nedir? Deneyiniz;

```
char str[255];
```

```
scanf("%s", str);
```

```
fgets(str, sizeof(str), stdin);
```



# Değer Aktarımında Pointerlar

Fonksiyonun geleneksel tanımlamasında bir fonksiyon ancak bir anda sadece bir değer döndürebilmektedir.

Ancak Bazen fonksiyonların, kendilerine parametre olarak geçilen değişkenler üzerinde doğrudan değişiklik yapması kullanışlı olabilmektedir.

Örneğin, `takas(int a, int b)` fonksiyonu, `a` ve `b` değerlerinin yerlerini değiştirsin istersek bu ancak ve ancak parametre olarak pointer kullanılarak mümkün olur.





# Struct Pointerları

Tüm ön tanımlı veri tiplerine işaretçi tanımlanabileceği gibi, programcı tanımlı struct yapılarına da pointer tanımlanabilmektedir;

```
typedef struct kayıt_t {  
    char adsoyad[80];  
    char tcno[11]  
    float puan;  
};
```

```
kayıt_t kayıtlar;  
kayıt_t *pkayıtlar;
```



# Pointer Pointerları

Pointer, bir değişkenin adresini tuttuğu gibi, Bu adrese işaret eden başka bir pointer daha tanımlanabilir.

İşaretçiyi işaret eden bu işaretçiler, \*\* iki pointer tanımlayıcısı ile tanımlanır.

```
int i, *p, **q;
```

```
i = 5;
```

```
p = &i;
```

```
q = &p;
```

```
printf("%d", **q);
```