1.Ders : C İşaretçiler (Pointers)

C İşaretçiler

Bu eğitimde, işaretçiler hakkında bilgi edineceksiniz; işaretçiler nelerdir, bunları nasıl kullanırsınız ve onlarla çalışırken karşılaşabileceğiniz genel hataları örnekler yardımıyla öğreneceksiniz.

İşaretçiler, C ve C ++ programlamanın güçlü özellikleridir. İşaretçileri öğrenmeden önce, C programlamadaki adresleri öğrenelim.

C'de Adres

Programınızda değişken var ise **&var** size hafızadaki adresi verecektir.

Scanf () işlevini kullanırken adresi kullandık.

```
scanf("%d", &var);
```

Burada kullanıcı tarafından girilen değer var değişkeninin adresinde saklanır. Çalışan bir örnek ele alalım.

```
#include <stdio.h>
int main()
{
  int var = 5;
  printf("var: %d\n", var);
  // Notice the use of & before var
  printf("address of var: %p", &var);
  return 0;
}
```

Çıktısı :

```
var: 5
address of var: 2686778
```

Not: Yukarıdaki kodu çalıştırdığınızda muhtemelen farklı bir adres alırsınız.

C İşaretçiler

İşaretçiler (işaretçi değişkenleri), değerler yerine adresleri saklamak için kullanılan özel değişkenlerdir.

İşaretçi Sözdizimi (Pointer Syntax)
İşaretçileri nasıl açıklayacağız? Böyle :

```
int* p;
```

Burada, int türünde bir **p işaretçisi** ilan ettik.

İşaretçileri bu yollarla da ilan edebilirsiniz.

```
int *p1;
int * p2;
```

İşaretçi bildirmek için başka bir örnek alalım.

```
int* p1, p2;
```

İşaretçilere adres atama

```
int* pc, c;
c = 5;
pc = &c;
```

Burada c değişkenine 5 atanmıştır. Ve, c adresi pc işaretçisine atanır.

İşaretçilerin gösterdiği değeri alma.

İşaretçilerin gösterdiği şeyin değerini almak için * operatörünü kullanırız. Örneğin:

```
int* pc, c;
c = 5;
pc = &c;
printf("%d", *pc);   // Çıktısı: 5
```

Burada, c adresi pc işaretçisine atanır. Bu adreste saklanan değeri elde etmek için * pc kullandık.

İşaretçilerin Gösterdiği Değeri Değiştirme

```
int* pc, c;
c = 5;
pc = &c;
c = 1;
printf("%d", c);    // Çıktısı: 1
printf("%d", *pc);  // Çıktısı: 1
```

C adresini pc işaretçisine atadık.

Daha sonra c'nin değerini 1 olarak değiştirdik. Pc ve c adresi aynı olduğundan, * pc bize 1 verir.

Başka bir örnek alalım.

```
int* pc, c;
c = 5;
pc = &c;
*pc = 1;
printf("%d", *pc);  // Çıktısı: 1
printf("%d", c);    // Çıktısı: 1
```

C adresini pc işaretçisine atadık.

Daha sonra * pc = 1; kullanarak * pc'yi 1'e değiştirdik. Pc ve c adresi aynı olduğundan, c 1'e eşit olacaktır.

Bir örnek daha alalım.

```
int* pc, c, d;
c = 5;
d = -15;
pc = &c; printf("%d", *pc); // Çıktısı: 5
pc = &d; printf("%d", *pc); // Çıktısı: -15
```

İlk olarak, c adresi, pc = & c; kullanılarak pc işaretçisine atanır. C değeri 5 olduğundan, * pc bize 5 verir.

Ardından, d adresi, pc = & d; kullanılarak pc işaretçisine atanır. D'nin değeri -15 olduğundan, * pc bize -15 verir.

Örnek: İşaretçi Nasıl Çalışır?

```
#include <stdio.h>
int main()
{
   int* pc, c;
   c = 22;
   printf("Address of c: %p\n", &c);
   printf("Value of c: %d\n\n", c);  // 22
   pc = &c;
   printf("Address of pointer pc: %p\n", pc);
   printf("Content of pointer pc: %d\n\n", *pc); // 22
   c = 11;
   printf("Address of pointer pc: %p\n", pc);
   printf("Content of pointer pc: %d\n\n", *pc); // 11
   *pc = 2;
   printf("Address of c: %p\n", &c);
   printf("Value of c: %d\n\n", c); // 2
   return 0;
}
```

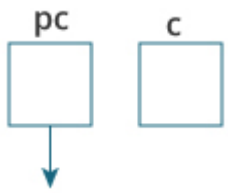
Çıktısı

```
Address of c: 2686784
Value of c: 22

Address of pointer pc: 2686784
Content of pointer pc: 22

Address of pointer pc: 2686784
Content of pointer pc: 11

Address of c: 2686784
Value of c: 2
```

Programın açıklaması

1)

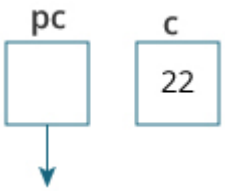
```
int* pc, c;
```



Burada, her iki int türünde bir işaretçi pc ve normal bir c değişkeni yaratılır.
Pc ve c başlangıçta başlatılmadığından, pointer pc herhangi bir adrese ya da rastgele bir adrese işaret etmez. Ve c değişkeni bir adrese sahiptir ancak rastgele çöp değeri içerir.

2)

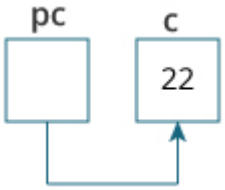
```
c = 22;
```



Bu, 22’yi **c** değişkenine atar. Yani, 22 **c** değişkeninin hafıza konumunda saklanır.

3)

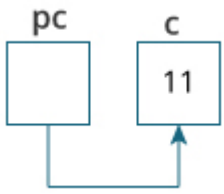
```
pc = &c;
```



Bu, c değişkeninin adresini pointer pc’ye atar.

4)

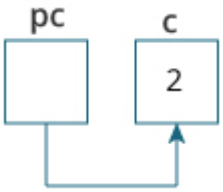
```
c = 11;
```



Bu, 11’i c değişkenine atar.

5)

*pc = 2;



Bu, işaretçi pc ile gösterilen hafıza konumundaki değeri 2 ile değiştirir.

İşaretçilerle çalışırken yapılan yaygın hatalar

Diyelim ki, işaretçi pc’nin c adresini göstermesini istiyorsunuz.

```
int c, *pc;
// Yanlış! pc adres ise,
// c bir adres değil.
pc = c;
// Yanlış! * pc, pc adresinde belirtilen değer ise,
```

```
// & c bir adres.
*pc = &c;
// Doğru! pc bir adres ve,
// & c ayrıca bir adres.
pc = &c;
// Doğru! * pc, pc adresinde saklanan bir değerdir ve,
// c aynı zamanda bir değerdir (adres değil).
*pc = c;
```

İşte işaretçi sözdizimi yeni başlayanlara bir örnek, biraz kafa karıştırıcı.

```
#include <stdio.h>
int main()
{
   int c = 5;
   int *p = &c;
   printf("%d", *p);  // 5
}
```

İnt * p = & c; kullanırken neden hata almadık?

```
int *p;
```

Sadece işaretçiler oluşturmak için kullanılan sözdizimidir. P işaretçisinin gösterdiği değeri almaya çalışmıyoruz.

```
int *p = &c;
```

eşittir

```
int *p;
p = &c;
```

Bu karışıklığı önlemek için yukarıdaki ifadeyi bu şekilde yazabilirsiniz.

```
int* p = &c;
```

Şimdi işaretçilerin ne olduğunu biliyorsunuz, bir sonraki derste işaretçilerin dizilerle nasıl ilişkili olduğunu öğreneceksiniz.