C Dili - 8. Konuk

POINTER NEDIR?

Basitçe, pointer, bir adrestir. Bir değişken olmak yerine, bir değişkenin hafızadaki adresini taşıyan bir 'ok işareti' dir.

==========

```
main()                      /* Pointer kullanımı örneği */
{
   int index,*pt1,*pt2;
   index = 39;                       /* herhangi bir değer  */
   pt1 = &index;                    /* 'index' in adresi    */
   pt2 = pt1;
   printf("Deger simdi %d %d %d dir.\n",index,*pt1,*pt2);
   *pt1 = 13;           /* 'index' in değerine değişiklik yapalım */
   printf("değiştikten sonra ise %d %d %d\n",index,*pt1,*pt2);
}
```

==========

Su an için, programın index değişkenini ve iki tane astrisk ile baslayan terimlerin tanımlandığı yere bakmayın. Aslında astrisk denilen bu işarete, biz şimdilik 'yıldız' diyelim.

Programda ilk önce, index değişkenine 39 değerini atıyoruz. Bunun altındaki satırda ise, pt1'e tuhaf bir değer atanmasını görüyoruz - index değişkeni, ve önünde bir & ampersand işareti ile. Bu örnekte, pt1 ve pt2 pointer dir, ve index de basit bir değişkendir. Simdi bir problemle karşı karşıyayız. Bu programda pointer kullanılıyor, fakat nasıl kullanılacağını öğrenmedik.

Bu görecekleriniz biraz aklinizi karıştıracak, fakat bunları anlamadan geçmeyin.

IKI ONEMLI KURAL

1. önüne ampersand işareti konmuş bir değişken, o değişkenin adresini belirtir. Yani altıncı satir, söyle okunabilir: "pt1, index isimli değişkenin adresini alır."

2. önüne yıldız konmuş bir pointer, kendisinin tuttuğu adreste bulunan değeri gösterir. Programın dokuzuncu satiri, söyle okunabilir: "pt1 pointer'inin gösterdiği yere, 13 değeri atandı."

HAFIZA YARDIMCISI

1. **&** 'i bir adres olarak düşünün.
2. ***** 'i adresteki değer olarak düşünün.

pt1 ve pt2 pointer olarak, kendileri bir değer taşımazlar, fakat bellekteki bir adresi gösterirler. Bu programda, 'index' değişkenini

gösteren pointer'lar olduğu için, değişkenin değerini hem index ile, hemde onun adresini taşıyan pointer'larla değiştirebiliriz.

Dokuzuncu satırda, index değişkeninin değeri, pt1 pointer'i ile değiştiriliyor. Program içinde 'index' i kullandığımız herhangi bir yerde, (pt1 başka bir şeye atanıncaya kadar), '*pt1' i de kullanmamız mümkündür, çünkü pt1, index'in adresini taşımaktadır.

BIR BASKA POINTER

Programa değişiklik katmak için, bir başka pointer daha tanımladım. "pt2" isimli bu pointer, yedinci satırda "pt1"'in taşıdığı adresi almaktadır. Bu atamadan önce, ayni henüz değer atanmamış değişkenler gibi içinde rastgele bilgiler vardır. Bundan sonra, "pt2" de "index" değişkeninin adresini taşımaktadır. örneğin, dokuzuncu satırda "*pt1" i "*pt2" ile değiştirsek de, sonuç ayni olacaktır - çünkü iki pointer da ayni adresi taşımaktadır.

SADECE BİR DEĞİŞKEN

Bu programda üç tane değişken var gibi görünse de, aslında bir tane değişken tanımlıdır. İki pointer ise, bu değişkenin adresini tutmaktadır. Bu durum, "printf" komutunun hep 13 değerini yazmasından da anlaşılabilir. Bu gerçekten anlaması zor bir kavramdır, fakat en küçük C programları dışında hepsi tarafından kullanıldığı için, öğrenmeniz gereklidir.

POINTER NASIL TANIMLANIR

Programın üçüncü satırında, ilk önce "index" isimli değişken tanımlanır, daha sonra da iki tane pointer tanımlaması göreceksiniz. İkinci tanım, su şekilde okunabilir: "pt1'in göstereceği adres, bir tamsayi değişkenine ait olacak." Yani, "pt1", tamsayi bir değişkeninin pointer'i olur. Ayni şekilde, "pt2" de, yine bir tamsayi değişkeninin pointer'i olur.

Bir pointer, bir değişkenin adresini taşımak için tanımlanır. tanımlandığından başka bir değişken tipi için kullanımı "uyumsuz veri tipi" hatasının oluşmasına sebep olur. örneğin, "float" tipi bir pointer, "int" tipli bir değişkenin adresini alamaz.

POINTER'LI IKINCI PROGRAMIMIZ

POINTER2.C:

```
================================================================
#include <stdio.h>
void main()
{
   char baslik[40],*orada,bir,iki;
   int *pt,list[100],index;
   strcpy(baslik,"Bu bir karakter dizisidir.");

   bir = baslik[0];                 /* bir ve iki ayni değeri taşırlar */
   iki = *baslik;
   printf("İlk çıktı %c %c\n",bir,iki);
```

```
   bir = baslik[8];               /* bir ve iki ayni değeri taşırlar */
   iki = *(baslik+8);
   printf("İkinci çıktı %c %c\n",bir,iki);
   orada = baslik+10;        /* baslik+10 ve baslik[10] aynidir. */
   printf("üçüncü çıktı %c\n",baslik[10]);
   printf("dördüncü çıktı %c\n",*orada);

   for (index = 0;index < 100;index++)
      list[index] = index + 100;
   pt = list + 27;
   printf("Besinci çıktı %d\n",list[27]);
   printf("altıncı cikti %d\n",*pt);
}
   ============================================================
```

Bu programda, iki tane pointer, iki tane dizi ve üç tane değişken tanımlıyoruz. "orada" isimli pointer, karakter tipi, ve "pt" ise, tamsayi tipindedir.

BIR DİZİ (Array) DEGISKENI ASLINDA BIR POINTER DIR

C programlama dilinde, bir dizi değişkeni, o stringin başlangıcını gösteren bir pointer olarak tanımlanmıştır. Programda bir bakin: önce "baslik" isimli diziye sabit bir string atıyoruz. Daha sonra, "bir" isimli değişkene, "baslik" in ilk harfini atıyoruz. Sonra, "iki" isimli değişkene, ayni değeri atıyoruz. İkinci satırda "*baslik[0]" yazmak yanlış olurdu, çünkü yıldız işareti, köşeli parantezlerin yerini almaktadır.

"baslik" i neredeyse tam bir pointer gibi kullanabilirsiniz, yegane farkı, tuttuğu adres değiştirilemez, ve daima o baslikın başlangıç adresini gösterir.

on ikinci satıra gelince, string'in dokuzuncu karakterinin (sıfırdan başladığımız için), iki ayrı şekilde "bir" ve "iki" isimli değişkenlere atandığını görüyoruz.

C programlama dili, pointer'in tipine göre, index ayarlamasını otomatik olarak yapar. Bu durumda, "baslik" bir "char" olarak tanımlandığı için, başlangıç adresine 8 eklenir. Şayet "baslik" "int" (tamsayi) olarak tanımlanmış olsa idi, index iki ile çarpılıp, "baslik" in başlangıç adresine eklenirdi.

"orada" bir pointer olduğu için, 16. satırda "baslik" in 11. elemanının adresini taşıyabilir. "orada" gerçek bir pointer olduğu için, herhangi bir karakter değişkeninin adresini gösterebilir.

POINTER VE ARITMETIK

Her çeşit işlemler, pointer'lar ile mümkün değildir. Pointer bir adres olduğundan, ona bir sabit rakam ekleyip, daha ilerideki bir adrese erişmek mümkündür. Ayni şekilde, pointer'in adresinde bir rakam çıkartıp, daha önceki hafıza bölgelerine erişmek mümkündür. İki pointer'i toplamak
pek mantıklı değildir, çünkü bilgisayardaki adresler sabit değildir. çıkacak rakamın tuhaf olacağı için pointer ile çarpma da yapılamaz. Ne yaptığınızı düşünürseniz, yapabilecekleriniz ve yapamayacaklarınız kendini belli edecektir.

TAMSAYI POINTER'I

"list" isimli tamsayi dizisine, 100 den 199 a kadar değerler verilir. Daha sonra, 28. elemanın adresini, "pt" isimli pointer'a atıyoruz. Daha sonra ekrana yazdığımızda, gerçekten de, o değeri aldığını görüyoruz.

Daha önceki konularda, bir fonksiyondan veri değerlerini döndürmek için iki metot olduğunu söylemiştim. İlki, bir dizi kullanarak idi. İkincisini herhalde tahmin edersiniz. Şayet tahmininiz "pointer sayesinde" idi ise, tebrikler.

```
CIFTYON.C:
==========================================================
main()
{
   int cevizler,elmalar;

   cevizler = 100;
   elmalar = 101;
   printf("başlangıç değerleri %d %d\n",cevizler,elmalar);

   /* "değiştir" i çağırınca,         */
   degistir(cevizler,&elmalar);  /* cevizlerin DEGERI ve,     */
                                       /* elmaların ADRESİNİ geçiriyoruz  */

   printf("Bitiş değerleri ise, %d %d dir..\n",cevizler,elmalar);
}
degistir(kuru_yemis,meyvalar)       /* kuru_yemis tamsayidir   */
int kuru_yemis,*meyvalar;           /* meyvalar bir tamsayi pointer'idir */
{
   printf("Değerler %d %d\n",kuru_yemis,*meyvalar);
   kuru_yemis = 135;
   *meyvalar = 172;
   printf("Sonraki degerler %d %d\n",kuru_yemis,*meyvalar);
}
```

==========

Burada, iki tane tamsayi değişkeni (pointer degil) tanımlıyoruz: "cevizler" ve "elmalar". Önce bunlara birer değer atıyoruz, ve "degistir" isimli fonksiyonu çağırıyoruz. çağırırken, "cevizler" in degeri (100), ve "elmalar" değişkeninin adresini geçiriyoruz. Fakat, fonksiyona da, bir değer ve bir adres geleceğini haber vermemiz gereklidir. Bunun için, fonksiyonun parametreleri tanımlanırken, bir adres taşıyacak olan sembolün başına bir yıldız koymamız yeterlidir.

Fonksiyonun içinde, bu iki değeri değiştirip, eski ve yeni değerleri ekrana yazıyoruz. Bu program çalıştığında, ana programdaki "cevizler" in değerinin ayni kaldığını fakat "elmalar" in yeni değerlerini aldığını
göreceksiniz.

"cevizler" in değerinin ayni kalmasının nedeni, fonksiyona bir değer geçirildiğinde, C dilinin o değerin bir kopyasını fonksiyona geçirmesi yüzündendir. Programa geri döndüğünüzde, değerin bir kopyasını
kullandığımız için asil değerin değişmediğini göreceksiniz.

"elmalar" in değerinin değişmesi ise, yine fonksiyona "elmalar" değişkeninin adresinin bir kopyası geçirildiği halde, bu adres ana programdaki "elmalar" a karşılık geldiği için, fonksiyonda bu adresteki
değeri değiştirir değiştirmez, "elmalar" in da değeri değişmiş olur.

==========

KARISIKPTR.C:

```
#include <stdio.h>

int main(void)
{
    char ch = 'c';
    char *chptr = &ch;

    int i = 20;
    int *intptr = &i;

    float f = 1.20000;
    float *fptr = &f;

    char *ptr = "Ben bir stringim";

    printf("\n [%c], [%d], [%f], [%c], [%s]\n",
                *chptr, *intptr, *fptr, *ptr, ptr);
}
```

==========

Burada da birçok farklı pointer tanımlayıp bunları kullanıyoruz. Bu programın çıktısı şöyle görünecektir:

[c], [20], [1.200000], [B], [Ben bir stringim]

==========

SWAP.C

```
#include <stdio.h>
void swap(int * q,int * p)
{
  int temp = *p;
  *p = *q;
  *q = temp;
}

int main()
{
  int a = 10, b = 2, x = 3, y = 5;
  printf("a,b,x,y: %d,%d,%d,%d\n", a, b, x, y);
  swap(&x, &y);
  swap(&a, &b);
  printf("a,b,x,y: %d,%d,%d,%d\n", a, b, x, y);
}
```

==========

Burada da bir fonksiyona 2 pointer geçiriyor ve bu fonksiyon sayesinde iki değişkenin değerlerini birbirleriyle değiştiriyoruz.
bu programın da çıktısı:
a,b,x,y: 10,2,3,5
a,b,x,y: 2,10,5,3
Şeklinde olacaktır.

ALANHESABI.C:

```
================================================================

#include <stdio.h>
void dortgen(int a, int b, int *alan, int *cevresi);

int main()
{
  int x, y;
  int alan, cevresi;
  printf("Boşlukla ayrılmış iki değer giriniz: " );
  scanf("%d %d", &x, &y);
  dortgen(x, y, &alan, &cevresi);
  printf("Alanı %d ve çevresi %d dir\n", alan, cevresi);
}

void dortgen(int a,int b, int *alan,int *cevresi)
{
  *alan = a * b;
  *cevresi = 2 * (a + b);
}
================================================================
```

Burada kullanıcıdan 2 değer aldıktan sonra bunlarla bir dörtgenin alanını ve çevresini hesaplayıp kullanıcıya sonucu sunuyoruz.
Hesaplama işlemi bir fonksiyon tarafından yapılıyor. Dikkat ederseniz fonksiyona 4 parametre geçiriyoruz. Bunlardan ikisi
değer olarak diğer ikisi de adres olarak gönderiliyor. Adres olarak göndermemizin sebebi, fonksiyonun bu iki parametrenin değerini, ana programda değiştirebilmesidir.