

Doğu Akdeniz Üniversitesi
Bilgisayar ve Teknoloji Yüksek Okulu
Bilgisayar Programcılığı Bölümü

BTEP 102 – Veri Yapıları ve Programlama

Bölüm 3: Göstergeler



R. KANSOY

1

Konu Başlıkları

4. Göstergeler

- 4.1 Gösterge Kavramı
- 4.2 Göstergelerin Bildirimi
- 4.3 Değişkenlerin Adresi
- 4.4 Göstergeye Adres Atama
- 4.5 Gösterge Aritmetiği
- 4.6 Dizilerle Göstergelerin Birlikte Kullanımı
- 4.7 Katar Sabitler İçin Göstergelerin Kullanımı
- 4.8 Gösterge Dizilerinin Yaratılması

BTEP 102 – Veri Yapıları ve Programlama

2

2

4. Göstergeler

- ✓ **Gösterge**, bir değişkenin bellekteki adresini tutan başka bir değişkendir.
- ✓ Örneğin, **b** değişkenin bellekteki konumunu, yani adresini gözönüne alalım. Bu adresi bir başka **a** değişkeni içine yerleştirelim. Bu durumda "**a, b'nin göstergesidir**" yada "**a, b'ye işaret etmektedir**" denir.
- ✓ Gösterge, bir değişkenin değerini değil, söz konusu değişkenin bellek üzerindeki adresini içermektedir.

3

3

4.1 Gösterge Kavramı

Gösterge Değişken

13b0fff6

↑

Gösterge değişken, bir başka değişkenin bellek üzerindeki adresini içeriyor.

Değişken

↑

Bu değişkenin bellek üzerindeki adresi 13b0fff6'dır.

Şekil 4.1: Gösterge değişkenin bellek üzerindeki adresini simgeler.

a gösterge değişkeni

13b0fff6

↑

a değişkeni, bir b değişkenin bellek üzerindeki adresini içeriyor.

b değişkeni

100

↑


b değişkeninin bellek üzerindeki adresi 13b0fff6'dır ve 100 değerini içermektedir.

Bir b değişkeninin adresi 13affff6 olsun. Bu adresi içeren değişken a ise, a b'nin göstergesidir.

Şekil 4.2: Bir a gösterge değişkeni b değişkeninin adresini içeriyor.

4

4




4.2 Göstergelerin Bildirimi

- ✓ Bir gösterge, diğer değişkenler gibi, sayısal bir değişkendir. Bu sebeple kullanılmadan önce program içinde bildirilmelidir.
- ✓ Gösterge tipindeki değişkenler şöyle tanımlanır:
 - `tür *değişken_adı;`
- ✓ `tür`, göstergenin tipini belirler. Örneğin tamsayı bir gösterge değişken için `int` bildirimi yapılır.
- ✓ Değişken isimlerinin başında `'*'` işlecine yer verilir. Bu işleç, gösterge değişkenlerinin tanımlanmasında kullanılır.
- ✓ Aşağıda tanımlanan gösterge değişkenlerden, `a` bir karakterin, `x` bir tamsayının ve `sonuc` bir gerçel sayının bellekte saklı olduğu yerlerin adreslerini tutar.
 - `char *a;`
 - `int *x;`
 - `float *sonuc;`

BTEP 102 – Veri Yapıları ve Programlama

5

5




4.3 Değişkenlerin Adresi

- ✓ Bildirimi yapılmış bir gösterge değişken herhangi bir şey ifade etmez.
- ✓ Bu göstergenin kullanılabilmesi için, söz konusu gösterge değişkene bir başka değişkenin adresini yerleştirmek gerekmektedir.
- ✓ Bir değişkenin bellek üzerindeki adresini öğrenmek için `"&"` işlecinden yararlanılır.
- ✓ Örneğin, `a` değişkeninin bellek üzerindeki adresini, `&a` biçiminde gösterebiliriz.
- ✓ Bu tür gösterge veya bir başka deyişle adres bilgilerinin görüntülenmesinde `printf()` deyimi içinde biçimlendirme ifadesi olarak `%p` kullanılır.
- ✓ Eğer adresin onaltılık (hexadecimal) düzende görüntülenmesi isteniliyorsa `%x` tanımı kullanılır.

BTEP 102 – Veri Yapıları ve Programlama

6

6



4.3 Değişkenlerin Adresi

a değişkeni

100

→

&a

→

13a5fff6

a değişkeni adresi


Şekil 4.3: & İşlecinin Görevi.

KOD 4.1 Bir Değişkenin Bellek Adresini Öğrenmek

```
#include <stdio.h>
main()
{
    int a=100;
    printf("a değişkeninin değeri: %d \n", a);
    printf("a değişkeninin bellek adresi: %p", &a);
}
```

Sonuç: a değişkeninin değeri:100
a değişkeninin bellek adresi: 13a5fff6

7



4.3 Değişkenlerin Adresi

KOD 4.2 Değişkenlerin Adres Bilgilerini Bulmak

```
#include <stdio.h>
main()
{
    int a,b=100;
    double x;
    int z;
    printf(" Adres a = %x \n",&a);
    printf(" Adres b = %x \n",&b);
    printf(" Adres x = %x \n",&x);
    printf(" Adres z = %x \n",&z);
}
```

Sonuç: Adres a=fff6
Adres b=fff4
Adres x=ffec
Adres z=ffea

8



4.4 Göstergeye Adres Atama

- ✓ Bir pointera, bir değişkenin adresini atamak için & adres-operatörü kullanılır.
- ✓ Bu operatör bir değişkenin önüne konursa, o değişkenin içeriği ile değilde adresi ile ilgileniliyor anlamına gelir.

KOD 4.3 Göstergenin İşaret Ettiği Değişkenin Değerini ve Bellek Adresini Elde Etmek	Sonuç:
<pre>#include <stdio.h> main() { int x, *isaret; x = 888; isaret = &x; printf("x in degeri = %d\n",x); printf("x in adresi = %x\n",isaret); printf("x in degeri = %d\n",*isaret); printf("x in adresi = %x\n",&x);} </pre>	<pre>x in degeri = 888 x in adresi = ffde x in degeri = 888 x in adresi = ffde </pre>

9

9




4.4 Göstergeye Adres Atama

KOD 4.4 Bir Değişkenin İçeriğini ve Adresini Ekrana Yazdırma	Sonuç:
<pre>#include <stdio.h> main() { int *ptam, tam = 33; ptam = &tam; /* ptam -> tam */ printf("&tam = %p\n",&tam); printf("ptam = %p\n",ptam); printf("\n"); printf("tam = %d\n",tam); printf("*ptam = %d\n",*ptam); printf("\n"); *ptam = 44; /* tam = 44 anlamında */ printf("tam = %d\n", tam); printf("*ptam = %d\n",*ptam); } </pre>	<pre>&tam = 0x3fffd14 ptam = 0x3fffd14 tam = 33 *ptam = 33 tam = 44 *ptam = 44 </pre>

10

10




4.5 Gösterge Aritmetiği

- ✓ Göstergeler kullanılırken, göstergenin gösterdiği adres taban alınıp, o adresten önceki veya sonraki adreslere erişilebilir.
- ✓ Bu durum, göstergeler üzerinde, aritmetik işlemcilerin kullanılmasını gerektirir.
- ✓ Göstergeler üzerinde yalnızca toplama (+), çıkarma (-), bir arttırma (++) ve bir eksiltme (--) operatörleri işlemleri yapılabilir.

11

11




4.5 Gösterge Aritmetiği

4.5.1 Değişken Adresini Artırma ve Eksiltme

KOD 4.5 + ve - Operatörlerini Kullanarak Değişken Adreslerinin Bulunması	Sonuç:
<pre>#include <stdio.h> main() { a= 500; printf("Adres a: %p\n",&a); printf("Adres a+1: %p\n",&a+1); printf("Adres a-1: %p\n",&a-1); }</pre>	<pre>Adres a: 13aefff6 Adres a+1: 13aefff8 Adres a-1: 13aefff4</pre>

12

12



4.5 Gösterge Aritmetiği


4.5.1 Değişken Adresini Artırma ve Eksiltme

KOD 4.6 Değişken Adreslerinin Göstergeler Kullanarak Elde Edilmesi	Sonuç:
<pre>#include <stdio.h> main() { a= 500; int *b; b=&a; printf("Adres a: %p\n", b); printf("Adres a+1: %p\n",b+1); printf("Adres a-1: %p\n",b-1); }</pre>	<pre>Adres a: 13aefff6 Adres a+1: 13aefff8 Adres a-1: 13aefff4</pre>

BTEP 102 – Veri Yapıları ve Programlama

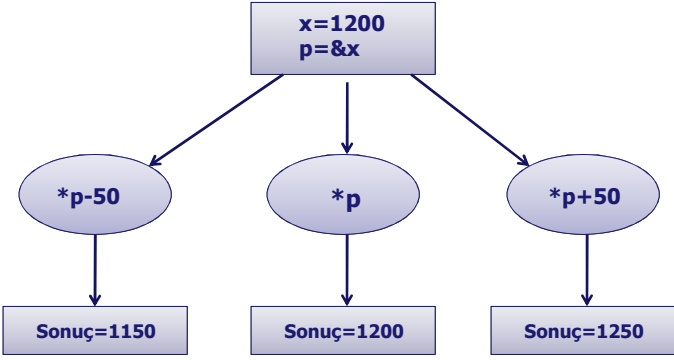
13

13



4.5 Gösterge Aritmetiği

4.5.2 Gösterge Değerini Artırma ve Eksiltme



```


graph TD
    A["x=1200  
p=&x"] --> B("p-50")
    A --> C("p")
    A --> D("p+50")
    B --> E["Sonuç=1150"]
    C --> F["Sonuç=1200"]
    D --> G["Sonuç=1250"]
  
```

Şekil 4.4: Gösterge Değeri Artırılarak Değişik Sonuçların Elde Edilmesi.

BTEP 102 – Veri Yapıları ve Programlama

14

14




4.5 Gösterge Aritmetiği

4.5.2 Gösterge Değerini Artırma ve Eksiltme

KOD 4.7 Gösterge Değeri 1 Artırılarak/Eksiltilerek Değişken Değerinin Görüntülenmesi	Sonuç:
<pre>#include <stdio.h> main() { a= 500; int *p; p=&a; printf("Sonuc 1: %d\n", *p); printf("Sonuc 2: %d\n", *p+1); printf("Sonuc 3: %d\n", *p-1); }</pre>	<p>Sonuc 1: 500 Sonuc 2: 501 Sonuc 3: 499</p>

15

15




4.5 Gösterge Aritmetiği

4.5.2 Gösterge Değerini Artırma ve Eksiltme

KOD 4.8 Değişken Değerlerini Göstergeleri Kullanmadan Elde Etmek	Sonuç:
<pre>#include <stdio.h> main() { a= 500; printf("Sonuc 1: %d\n", a); printf("Sonuc 2: %d\n", a+1); printf("Sonuc 3: %d\n", a-1); }</pre>	<p>Sonuc 1: 500 Sonuc 2: 501 Sonuc 3: 499</p>

16

16



4.5 Gösterge Aritmetiği


4.5.3 Gösterge İşlemlerinde ++ ve -- İşleçlerinin Kullanımı

ÖRNEK: x, y ve z değişkenlerini göz önüne alalım. Bunlardan z değişkenine bir *gösterge göstergesinin işaret ettiğini varsayalım. Göstergeyi birer birer artırarak diğer değişkenlerin adreslerine ulaşabiliriz.

KOD 4.9 Gösterge İşlemlerinde ++ İşlecinin Kullanımı	Sonuç:
<pre>#include <stdio.h> main() { double x= 5.3; double y= 11.7; double z= 3.3; double *gösterge = &z; printf("Adres= %x Degeri = %4.1f\n", gosterge, *gösterge); gosterge++; printf("Adres= %x Degeri = %4.1f\n", gosterge, *gösterge); gosterge++; printf("Adres= %x Degeri = %4.1f\n", gosterge, *gösterge); gosterge++; }</pre>	<p>Adres : ffe0 Degeri = 3.3 Adres : ffe8 Degeri = 11.3 Adres : fff0 Degeri = 5.3</p>

17

17




4.5 Gösterge Aritmetiği

4.5.3 Gösterge İşlemlerinde ++ ve -- İşleçlerinin Kullanımı

KOD 4.10 ++ ve -- İşleçlerinin Göstergelerle Kullanımı	Sonuç:
<pre>#include <stdio.h> main() { a= 500; int *p; p=&a; printf("Sonuc 1: %d\n", *p); printf("Sonuc 2: %d\n", ++(*p)); printf("Sonuc 3: %d\n", --(*p)); }</pre>	<p>Sonuc 1: 500 Sonuc 2: 501 Sonuc 3: 499</p>

18

18



4.5 Gösterge Aritmetiği


4.5.3 Gösterge İşlemlerinde ++ ve -- İşleçlerinin Kullanımı

KOD 4.10'da yer alan bazı işlemleri şu şekilde yorumlayabiliriz.

- ***p** a değişkeninin içeriğini verir. *p göstergesi 500 değerini alır.
- **++(*p)** *p gösterge değerinin 1 fazlasını verir. Bu durumda *p göstergesi $500 + 1 = 501$ değerini alır.
- **--(*p)** *p gösterge değerinin 1 eksikliğini verir. Bu durumda *p göstergesi $500 - 1 = 499$ değerini alır.

19

19




4.5 Gösterge Aritmetiği

4.5.3 Gösterge İşlemlerinde ++ ve -- İşleçlerinin Kullanımı

KOD 4.11 Artırma ve Eksiltme İşleçlerinin Göstergeler İçin Kullanımı	Sonuç:
<pre>#include <stdio.h> main() { a= 500; int *p; p=&a; printf("Sonuc 1: %p\n", p); printf("Sonuc 2: %p\n", ++p); printf("Sonuc 3: %p\n", --p); }</pre>	<p>Sonuc 1: 13affff6 Sonuc 2: 13affff8 Sonuc 3: 13affff4</p>

20

20



4.5 Gösterge Aritmetiği


4.5.4 İşlem Sonucunu Adrese Yerleştirme

BTEP 102 – Veri Yapıları ve Programlama

- ✓ C’de bir islemin sonucunu bir göstergeye atayabilmek mümkündür.
- ✓ Örneğin x ve y degiskenleri tanimlanmis ise, bu iki degiskenin toplami da bellek üzerinde bir yer tutacaktır.
- ✓ Islem sonucunda elde edilen deger ayri bir bellek adresine yerlestirilebilir.
- ✓ Gerek görüldüğünde, adres bilgisi kullanılarak dolu bir bellek alanina da atama yapılabilir.
- ✓ Bu islemin sonucunun yer aldığı bellek adresi bir göstergeye atanabilir.

21

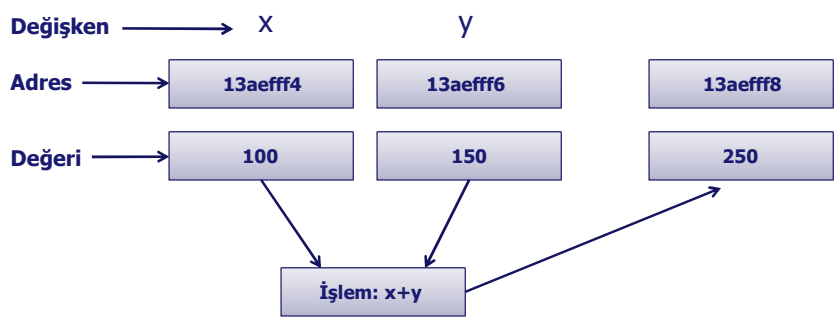
21



4.5 Gösterge Aritmetiği

4.5.4 İşlem Sonucunu Adrese Yerleştirme

BTEP 102 – Veri Yapıları ve Programlama




The diagram illustrates memory storage for variables X and Y, and the result of their addition. It shows three memory locations with addresses 13aefff4, 13aefff6, and 13aefff8. The first two locations contain the values 100 and 150 respectively, which are the values of variables X and Y. The third location contains the value 250, which is the result of the operation x+y. Arrows indicate the flow of data from the first two locations to the third, and from the third location to a box labeled 'İşlem: x+y'.

22

Şekil 4.4: İşlem Sonucunun Bir Diğer Adrese Yerleştirilmesi.

22




4.5 Gösterge Aritmetiği

4.5.4 İşlem Sonucunu Adrese Yerleştirme

KOD 4.12 İki Değişkenin Değeri Toplanarak Bir Gösterge ile Belirlenen Yere Yerleştiriliyor	Sonuç:
<pre>#include <stdio.h> main() { int x=4; int y=2; int z=5; int *gosterge; gosterge = &z; *(gosterge+2) = y+z; printf("x= %d\n", x); printf("y= %d\n", y); printf("z= %d\n", z); }</pre>	<pre>x=7 y=2 z=5</pre>

23

23



4.5 Gösterge Aritmetiği


4.5.5 * ve & İşleçlerinin Birlikte Kullanımı

- ✓ Gösterge işlemlerinde * ve & işleçlerini birlikte kullanmak mümkündür.
- ✓ Ancak bu işleçler birbirinin tersi işlemleri yerine getirdiği için, birbirlerini etkisizleştirirler.
- ✓ Örneğin *&a biçimindeki bir tanımda, * ve & işleçleri birbirini götürdüğü için, bu işleçlerin sonuca bir etkisi olmaz ve doğrudan doğruya a değişkeninin içeriği elde edilir.

KOD 4.13 * ve & İşleçlerinin Birlikte Kullanımı	Sonuç:
<pre>#include <stdio.h> main() { a= 500; int *p; p=&a; printf("Sonuc 1: %d\n", p); }</pre>	<pre>Sonuc : 500</pre>

24

24




4.6 Dizilerle Göstergelerin Birlikte Kullanımı

- ✓ C dilinde gösterge ve diziler arasında yakın bir ilişki vardır.
- ✓ Bir dizinin adı, dizinin ilk elemanının adresini saklayan bir göstergedir.
- ✓ Bu adrese dizinin temel adresi denilir.
- ✓ Dizinin temel adresini bulmak için, “&” islecinden yararlanılır.
- ✓ Örneğin, bir `a[]` dizisinin temel adresi `&a[0]` biçiminde elde edilir.
- ✓ Bir dizi indeksiz olarak kullanılırsa, bu dizi adı da dizinin temel adresini verir.
- ✓ Örneğin, `a[]` dizisinin temel adresini bulmak için sadece `a` yazmak da yeterlidir.

BTEP 102 – Veri Yapıları ve Programlama

25

25



4.6 Dizilerle Göstergelerin Birlikte Kullanımı

$a[i]$	$a[0]$	$a[1]$	$a[2]$	$a[3]$	$a[4]$
Değerler	2	7	0	3	9
Adresler	0022FF44	0022FF48	0022FF4C	0022FF50	0022FF54

↑
Dizinin temel adresi


Şekil 4.5: Dizinin İçerdiği Değerler ve Adresleri.

BTEP 102 – Veri Yapıları ve Programlama

26

26

BTEP 102 – Veri Yapıları ve Programlama



4.6 Dizilerle Göstergelerin Birlikte Kullanımı


- ✓ Örneğin: `int kutle[5], *p, *q;` şeklinde bir bildirim yapılsın.
- ✓ Buna göre aşağıda yapılan atamalar geçerlidir:
 - `p = &kutle[0];` /* birinci elemanın adresi p göstergesine atandı */
 - `p = kutle;` /* birinci elemanın adresi p göstericisine atandı */
 - `q = &kutle[4];` /* son elemanın adresi q göstericisine atandı */
- ✓ Ayrıca, i bir tamsayı olmak üzere, `kutle[i];` ile `*(p+i);` aynı anlamdadır.
- ✓ `p+i` işlemi ile `i+1`. elemanın adresi, ve `*(p+i)` ile de bu adresteki değer hesaplanır.

```
*p+i; /* p nin gösterdiği değere (dizinin ilk elemanına) i sayısını ekle */
*(p+i); /* p nin gösterdiği adresten i blok ötedeki sayıyı hesapla */
Çünkü, * operatörü + operatörüne göre işlem önceliğine sahiptir.
```

27

27

BTEP 102 – Veri Yapıları ve Programlama




4.6 Dizilerle Göstergelerin Birlikte Kullanımı

ÖRNEK: Bir göstergenin bir `a[]` dizisine işaret etmesini sağlayarak, dizinin tüm elemanlarını görüntülemek istiyoruz.

KOD 4.14 Dizilerle Göstergelerin Birlikte Kullanımı	Sonuç:
<code>#include <stdio.h></code>	2
<code>main()</code>	7
<code>{</code>	0
<code>int a[5]= {2,7,0,3,9};</code>	3
<code>int *p;</code>	9
<code>int i;</code>	
 <code>p=a;</code>	
 <code>for (i=0;i<5;i++)</code>	
<code>printf(" %d\n", *(p+i));</code>	
<code>}</code>	

28

28



4.6 Dizilerle Göstergelerin Birlikte Kullanımı


4.6.1 Katarlar ve Göstergeler

- ✓ C 'de özel bir "karakter dizisi" (katar) tipi bulunmadığı için karakterlerden oluşan normal bir dizi ya da bir karakter göstergesi bir karakter dizisi olarak düşünülebilir.
- ✓ Bir karakter dizisinin, bir gösterge yardımıyla kullanılabilmesi için, dizinin doğrudan doğruya göstergeye atanması yeterlidir.
- ✓ Bu işlemin ardından, gösterge indeksli bir dizi gibi kullanılabilir.
- ✓ Örneğin bir ***g** göstergesi bir **a[]** katarına **g=a** biçiminde işaret edilebilir.

BTEP 102 – Veri Yapıları ve Programlama

29

29



4.6 Dizilerle Göstergelerin Birlikte Kullanımı

4.6.1 Katarlar ve Göstergeler


ÖRNEK: Bir p göstergesinin bir a[] dizisine işaret ettiğini varsayalım. Bu durumda, karakter dizisini yazdırmak için p göstergesinden yararlanılabilir.

KOD 4.15 Karakter Dizisini Yazdırmak İçin Göstergelerin Kullanımı	Sonuç:
<pre>#include <stdio.h> main() { char a[]= "Dogu Akdeniz Universitesi"; char *p; int i; p=a; for (i=0;p[i];i++) printf(" %c", p[i]); }</pre>	Dogu Akdeniz Universitesi

BTEP 102 – Veri Yapıları ve Programlama

30

30



4.7 Katar Sabitler için Göstergelerin Kullanımı


- ✓ C programlama dilinde katar sabitler, tirnak isaretleri arasında tanımlanır.
- ✓ Derleyici bu tür bir katar ile karsilastiginda, onu programin katar tablosu içinde saklar ve bu katar için bir gösterge üretir.
- ✓ Bu nedenle, C programi içinde bir göstergenin katar sabitlerine isaret etmesi saglanarak dogrudan kullanılabilir.

KOD 4.16 Katar Sabitlerin Gösterge Kullanılarak Yazdırılması	Sonuç:
<pre>#include <stdio.h> char *p= "Dogu Akdeniz Universitesi"; main() { printf(p); }</pre>	Dogu Akdeniz Universitesi

BTEP 102 – Veri Yapıları ve Programlama

31

31



4.8 Gösterge Dizilerinin Yaratılması

- ✓ Göstergelerden de herhangi bir veri türünde diziler yaratmak mümkündür.
- ✓ Örneğin, tamsayı türünde 10 elemanlı bir ***g** gösterge dizisi şu şekilde tanımlanır;
 - `int *g[10]`

KOD 4.17 Gösterge Dizilerinin Kullanımı	Sonuç:
<pre>#include <stdio.h> int i; char *p[] = {"Pazartesi", "Sali", "Carsamba", "Persembe", "Cuma", "Cumartesi", "Pazar"}; main() { for (i=0; *p[i]; i++) printf("%s\n", p[i]); }</pre>	Pazartesi Sali Carsamba Persembe Cuma Cumartesi Pazar

BTEP 102 – Veri Yapıları ve Programlama

32

32