

Diziler (Arrays)

Doç. Dr. Fatih ÖZYURT

Dizi (Array)

- Aynı tipe (ilkel tipler ya da kullanıcı tanımlı tipler) sahip veri elemanlarının (aynı türden nesnelerin) oluşturduğu veri yapılarına (ilişkili veri elemanları topluluğu) **dizi** denir.
- Diziler **statik** elemanlardır. Programın çalışma süresi boyunca sabit boyuttadırlar.
- Dizilerde **döngü işlemleri** sıklıkla kullanılır. Özellikle **for döngüleri** dizilerle kullanılmaya çok uygundur.

Fizikse Bellek Adresi	201	202	203	204	205	206	...	220
Veri								
İndis	0	1	2	3	4	5	...	19

Dizi (Array)

- Bir dizi içerisindeki bütün elemanlara aynı isimle ulaşılır. Yani dizideki bütün elemanların isimleri ortaktır. **Diziler için bellekte ardışık yer açılır.** Elemanlar arasındaki ayırt edici özellik, bellekteki yeridir. Dizi elemanlarına **indisler** ya da başka yöntemlerle doğrudan hızlı bir şekilde erişilebilir. ($O(1)$)
- **Dizi**, belirli sayıda ve aynı veri türünden değişkenlere aynı adla erişilmesini sağlayan bir yapıdır.

Fizikse Bellek Adresi	201	202	203	204	205	206	...	220
Veri								
Indis	0	1	2	3	4	5	...	19

Dizi (Array)

- Diziler **tek boyutlu**, **iki boyutlu** ya da **çok boyutlu** olarak sınıflandırılabilir.
- Diziler **tek boyutlu** olabileceği gibi **iki veya daha çok boyutlu** da olabilirler.

Matrisler iki boyutlu dizilere örnektir.

A	B	C	D	E	F	...	K
---	---	---	---	---	---	-----	---

	Sütun	
Satır	A	B
	C	D
	E	F
	...	K

Fizikse
Bellek
Adresi

201 202 203 204 205 206 ... 220

Veri

A	B	C	Ç	D	E	...	K
---	---	---	---	---	---	-----	---

Tek B.
İndis

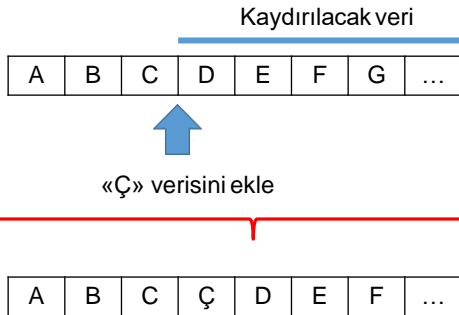
0 1 2 3 4 5 ... 19

İki B. İndis

0,0 0,1 1,0 1,1 2,0 2,1 ... 9,1

Dizi (Array)

- Sıralı dizi** içerisine eleman ekleme çıkarma işlemleri nispeten zordur. Elemanların kaydırılması gerekebilir.



1. Tek Boyutlu Dizi (Array)

- Oluşturma:**

```
veri_tipi dizi_adı[boyut];
```

```
int Sayilar[5];
```

- Oluşturma ve İlk Değer Atama:**

```
veri_tipi dizi_adı[boyut] = {değer,değer,...};
```

```
int Sayilar[5] = {11,5,23,4,54};
```

- Dizi Elemanına Değer Atama:**

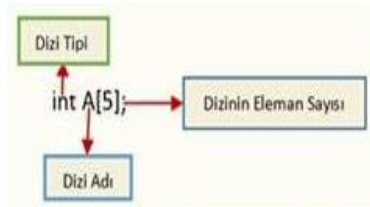
```
dizi_adı[indis] = değer;
```

```
Sayilar [2] = 77;
```

- Dizi Elemanının Değerine Erişim:**

```
değişken = dizi_adı[indis];
```

```
int a = Sayilar [1];
```



A[0]	1. Eleman	Dizinin İlk Elemanı (A[0])
A[1]	2. Eleman	
A[2]	3. Eleman	Dizinin Üçüncü Elemanı (A[2])
A[3]	4. Eleman	
A[4]	5. Eleman	Dizinin Son Elemanı (A[4])

Bağlı Listeler ve Dizilerin Karşılaştırma Tablosu

	Bağlı Liste (Linked List)	Dizi (Array)
1	Bağlı Listeler, her bir elemanın işaretçiler kullanılarak bir sonrakine bağlandığı aynı türdeki öğelerin sıralı bir koleksiyonudur.	Diziler, aynı tipteki veri türü elemanlarının bir koleksiyonudur.
2	Bağlı Listelerin elemanlarına sıralı olarak erişilebilir.	Dizilerin elemanlarına rastgele erişilebilir.
3	Bağlı Liste elemanları bellekte rastgele saklanır	Dizilerin elemanları, bellekte ardışık adreslerde saklanır.
4	Bağlı listelerde elemanların yerleştirilmesi ve silinmesi hızlı ve kolaydır.	Dizilerin elemanlarını yerleştirilmesi ve silinmesi maliyetlidir.
5	Bağlı Liste elemanları için bellek çalışma zamanı sırasında atanır (Dinamik Bellek Ayırma)	Dizi elemanları için bellek, derleme sırasında atanır (Statik Bellek Ayırma).
6	Bağlı listenin boyutu, yeni elemanlar eklendiğinde / silindiğinde büyür / küçülür.	Dizilerin boyutu, yeni elemanlar eklendiğinde / silindiğinde büyümeyebilir / küçülmez.
7	Bağlı listelerde doğrusal arama yapılabilir.	Dizilerde doğrusal ve ikili arama yapılabilir.
8	Bağlı listeler daha fazla bellek kullanır	Diziler daha az bellek kullanır

1. Tek Boyutlu Dizi (Array)

- Dizinin ilk elemanın indis değeri “0” son elemanın indis değeri ise “**boyut-1**” olur.
- `int Sayilar[];` şeklinde **boyutu belli olmayan** dizi tanımlanamaz, “**unknown size**” hatası alınır. Dizi boyutu program çalışma süresince sabittir.
- `int n[5] = { 2, 12, 13, 4, 5 };`
eleman sayısından az ise diğerleri 0 olur, fazla ise hata olur.
- `int n[5] = { 0 };`
hepsi 0 olur

`int c[12];`

c[0]	-45
c[1]	6
c[2]	0
c[3]	72
c[4]	1543
c[5]	-89
c[6]	0
c[7]	62
c[8]	-3
c[9]	1
c[10]	6453
c[11]	78

1. Tek Boyutlu Dizi (Array)

• Örnek:

Bir dizinin eleman değerlerini ekrana yazdırma;

```
int Sayilar[5]; //Dizi tanımladık
```

```
Sayilar[0]=11; //Dikkat! Dizilerde ilk elemanın numarası sıfırdır. Yani sıfırdan  
//başlar. Bilindik şekilde 1'den başlamaz...
```

```
Sayilar[1]=5;
```

```
Sayilar[2]=23;
```

```
Sayilar[3]=4;
```

```
Sayilar[4]=54; //Bu bizim beşinci (son) elemanımız. (Boyut-1)
```

//Yani bir dizinin boyutu beş ise son elemanın indeks değeri 4 olur.

//Dizinin ilk elemanı sıfırdan başladığı için son elemanın indeks değeri

//eleman sayısının bir eksiği olur.

1. Tek Boyutlu Dizi (Array)

- Diziler sıfırdan başlayan alt indexler kullanırlar
 - **ilk eleman**ın indexi **0**
 - **ikinci eleman**ın indexi **1**
 - **n. eleman**ın indexi **n-1**
 - **son eleman**ın indexi **length-1**
- Örnek:

```
int[] scores = {97, 86, 92, 71};
```

index:	0	1	2	3
deger:	97	86	92	71

Dizinin Elemanlarını Yazdırma

ALGORITHM Yazdır(A, n)

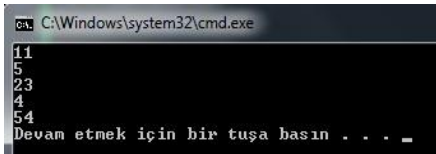
INPUT n adet tam sayıdan oluşan A dizisi

FOR sayaç \leftarrow 0 **TO** n - 1 **do**

PRINT A[sayaç]

RETURN 0

```
int sayac;  
for  
(sayac=0;sayac<5;sayac++)  
{  
    cout <<  
    Sayilar[sayac]<<endl;  
}
```



```
C:\Windows\system32\cmd.exe  
11  
5  
23  
4  
54  
Devam etmek için bir tuşa basın . . . _
```

1. Tek Boyutlu Dizi (Array)

- **Örnek:**

Bir dizinin eleman sayısını ekrana yazdırma;

```
int ElemanSayisi =  
sizeof(Sayilar)/sizeof(Sayilar[0]);
```

```
int Sayilar[5]={11,5,23,4,54};
```

```
int eleman_sayisi = (sizeof(Sayilar)/sizeof(int));
```

```
cout << "Dizinin Eleman Sayisi = " << eleman_sayisi<<  
std::endl;
```

1. Tek Boyutlu Dizi (Array)

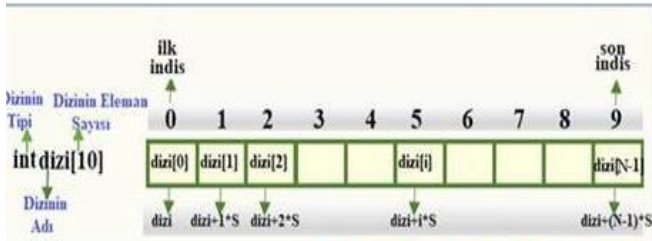
- ✓ Dizi elemanları bellek hücrelerine ardışık olarak yerleşir.
- ✓ Dizinin ismi dizinin ilk elemanının adresine karşılık gelir.
- ✓ Dizi elemanlarının tipi aynıdır.

Dizinin ilk elemanının adresinin bilinmesi ve dizi elemanlarının tiplerinin aynı olması bilgilerinden yararlanılarak dizinin diğer elemanlarının adresleri hesaplanabilir.

int dizi[10] şeklinde tanımlanmış bir dizi olduğunu varsayalım ve yukarıda verilenlerden yararlanarak bu dizinin herhangi bir elemanının adresini döndüren fonksiyon oluşturalım.

dizi	Başlangıç adresi
N	Dizinin eleman sayısı
S	Dizinin Bir elemanının bellekte kapladığı yer
Adres dizi[i]	<i>i</i> indisli elemanın bellek adresi

1. Tek Boyutlu Dizi (Array)



Genel olarak tek boyutlu bir dizinin bellek erişim fonksiyonu aşağıdaki şekilde ifade edilebilir:

$$(dizi, i, S) = (dizi + i * S)$$

Örnek olarak `int dizi[10]={11, 22, 33, 3, 7, 2, 4, 15, 13, 8}` şeklinde tanımlanan dizinin en son elemanının adresini yukarıda verilen fonksiyonu kullanarak hesaplayalım. En son elemanın indis (*i*) değeri $i=10-1=9$ olarak hesaplanır. Dizinin tipi integer olduğu için bellekte kapladığı alan 4 Byte 'tır. Yani $S=4$ 'tür. Dizinin ilk elemanının adresini `dizi=1000` olduğunu kabul ederek yukarıdaki rakamları fonksiyonda yerine koyup dizinin son elemanının adresini;

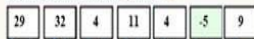
$$(dizi, i, S) = (1000 + 9 * 4)$$

$$= 1036$$

Doğrusal Arama

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <locale.h>
4 int main()
5 {
6     setlocale(LC_ALL, "Turkish");
7     int i, Aranan, kontrol=0;
8     int dizi[7]={29,32,4,11,4,-5,9};
9     printf("\nDizide Aranan Elemanı Giriniz : ");
10    scanf("%d", &Aranan);
11    for(i=0; i<6; i++)
12    {
13        if(Aranan==dizi[i]){
14            printf("\ndizinin %d . sıradaki elemanı Aranan Elemandır\n", i+1);
15            kontrol=1;
16        }
17    }
18    if (kontrol==0)
19        printf("\nAranan Eleman Bulunamadı\n");
20    return 0;
```

Dizide Aranan Elemanı Giriniz : 5



~~5~~×29 ~~5~~×32 ~~5~~×4 ~~5~~×11 ~~5~~×4 5×-5 ~~5~~×9

dizinin 6 . sıradaki elemanı Aranan Elemandır

Dizide Aranan Elemanı Giriniz : 4

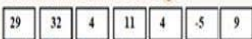


~~4~~×29 ~~4~~×32 4×4 ~~4~~×11 4×4 ~~4~~×-5 ~~4~~×9

dizinin 3 . sıradaki elemanı Aranan Elemandır

dizinin 5 . sıradaki elemanı Aranan Elemandır

Dizide Aranan Elemanı Giriniz : 8



~~8~~×29 ~~8~~×32 ~~8~~×4 ~~8~~×11 ~~8~~×4 ~~8~~×-5 ~~8~~×9

Aranan Eleman Bulunamadı

Program üç defa çalıştırılmış, klavyeden önce -5, sonra 4 ve daha sonrada 8 değerleri girilmiştir. Bu girişlerin sonunda ekrandan alınan çıktılar aşağıda sırasıyla gösterilmiştir.

2. İki Boyutlu Diziler

iki satır ve üç sütundan oluşan iki boyutlu bir dizi ***dizi[2][3]*** şeklinde tanımlanır ve ***dizi[i][j]*** şeklinde ifade edilir. Bu örnek için i değeri en fazla 2 (iki) ve j değerleri en fazla 3 (üç) değerini alabilir.

Tanımı yapılan bu matrisin N satırı ve M sütunu olduğu varsayılırsa, eleman sayısı $N \times M$ ile ifade edilir. Yukarıda verilen örnekte $N=2$ ve $M=3$ olduğu için matrisin eleman sayısı $2 \times 3 = 6$ olarak hesaplanır. Matrisin ilk elemanı ***dizi[0][0]***, son elemanı ise ***dizi[1][2]*** dir.

<u>dizi[0][0]</u>	*****	
<u>dizi[0][1]</u>	1. eleman	Dizinin 1. (ilk) Elemanı (dizi[0][0]
<u>dizi[0][2]</u>	2. eleman	
<u>dizi[1][0]</u>	3. eleman	
<u>dizi[1][1]</u>	4. eleman	
<u>dizi[1][2]</u>	5. eleman	
	6. eleman	Dizinin 6. (son) Elemanı (dizi[1][2]

2. İki Boyutlu Dizi Kullanarak Birim Matris Oluşturmak

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main() {
4      int dizi[4][4], i, j;
5      for(i=0; i<4; i++){
6          for(j=0; j<4; j++){
7              if(i==j){
8                  dizi[i][j]=1;
9                  printf("  %d", dizi[i][j]);
10             }
11             else
12             {
13                 dizi[i][j]=0;
14                 printf("  %d", dizi[i][j]);
15             }
16             printf("\n");
17         }
18     }
19     return 0;
20 }
```

Birim Matris(C programı Örneği)

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Birim Matris (Ekran Çıktısı)

3. Üç Boyutlu Diziler

Üç boyutlu diziler tanımlanırken 3 (üç) indis (*i, j, k*) kullanılır. Bu indislerden birinci indis yüzey bilgisini, ikinci indis satır bilgisini ve üçüncü indis sütun bilgisini gösterir. Üç boyutlu diziler bilgisayarda birçok probleme ait çözümün modellenmesinde önemli birer araçtır.

Örneğin bir dönemde 6 (altı) ders alan 50 öğrencinin her dersten aldığı notları bilgisayara girmek için ***float notlar[50][6][3]*** ; şeklinde bir dizi tanımlanabilir. Bu dizinin 900 (Dokuz Yüz) elemanı olduğuna dikkat ediniz.

3. Üç Boyutlu Diziler

1. Yöntem

`int dizi[3][3][3]`
`{11,22,33,3,6,9,8,10,12,24,25,26,27,28,29,30,31,32,34,35,36,37,38,39,40,41,42};`

2. Yöntem

`int dizi[3][3][3] = { { {11,22,33},{3,6,9},{8,10,12}},`
`{ {24,25,26},{27,28,29},{30,31,32}},`
`{ {34,35,36},{37,38,39},{40,41,42}}};`

4. İşaretçiler ve Diziler

Bir dizi tanımlanırken önce dizide yer alacak elemanların tipi, sonra dizinin ismi ve daha sonra da dizinin eleman sayısı(boyutu) belirtilir. Örneğin 5 elemanlı, tamsayı (int) tipindeki değerleri tutacak bir dizi aşağıdaki gibi tanımlanır.

```
int dizi[5];
```

Bu tanımlama, bellekte program için araka arkaya 4 Byte' lık 5 adet alan ayrılmasını sağlar. Bu alanların adreslerine adres operatörü ile ulaşılabilir.

Dizinin Elemanlarının Sırası	Dizinin Elemanları	Dizi Elemanlarının Adresi
İlk eleman	dizi[0]	&dizi[0]
İkinci eleman	dizi[1]	&dizi[1]
Üçüncü eleman	dizi[2]	&dizi[2]
Dördüncü eleman	dizi[3]	&dizi[3]
Beşinci Eleman	dizi[4]	&dizi[4]

4. İşaretçiler ve Diziler

İşaretçilerle diziler arasındaki ilişki, ***bir dizinin ismi, o dizinin ilk elemanının adresini tutan bir göstericidir*** şeklinde ifade edilebilir. Dolayısı ile bu tanımdan faydalanılarak dizi elemanlarının adresleri iki farklı şekilde yazdırılabilir. Tabloda dizi elemanlarının adreslerinin iki farklı şekilde nasıl yazdırılacağı ve dizi elemanlarının değerlerine, işaretçi yaklaşımı ile nasıl ulaşılabileceği gösterilmiştir.

Dizi Elemanlarının Sırası	Dizi Elemanlarının Adr. (1)	Dizi Elemanlarının Adr. (2)	Dizi Elemanl. Değeri
İlk Eleman	&dizi[0]	dizi	*dizi
İkinci Eleman	&dizi[1]	dizi + 1	*(dizi + 1)
Üçüncü Eleman	&dizi[2]	dizi + 2	*(dizi + 2)
Dördüncü Eleman	&dizi[3]	dizi + 3	*(dizi + 3)
Beşinci Eleman	&dizi[4]	dizi + 4	*(dizi + 4)

4. İşaretçiler ve Diziler

İşaretçiler değişkenleri işaret ettiği gibi benzer şekilde dizilere de işaret edebilirler.

Örneğin, `int dizi[5];` şeklinde tanımlanmış bir dizinin, işaretçi (pointer) ile işaret edilmesi istenirse, `ptr = dizi;` yazılması yeterlidir (Bkz. Program 5.8, 9. Satırdaki ifade). Değişkenlerde, değişken adının başına '&' işareti getiriliyordu, fakat dizilerde buna gerek yoktur.

Çünkü dizilerin kendisi de bir işaretçidir. Dizilerin hepsi bellekte bir başlangıç noktası işaret eder. Örnek olması açısından bellekte başlangıç noktasının 1000 olduğunu varsayalım. Bu durumda "`dizi[0]`" dendiği zaman 1000 ile 1004 arasında kalan bölgenin kullanılacağı anlaşılmalıdır. Ya da "`dizi[4]`" dediğiniz zaman 1020 ile 1024 bellek bölgesi işleme alınır.

Bir diziye işaret eden işaretçi (pointer) dizi gibi kullanılabilir. Yani `ptr = dizi;` ifadesi yazıldıktan sonra, **`ptr[0]` ile `dizi[0]`** birbirinin aynısıdır. Eğer `*ptr` yazılırsa, yine dizinin ilk elemanı `dizi[0]` işaret edilmiş olur. Ancak dizi işaret eden işaretçiler genellikle, `*(ptr + 0)` şeklinde kullanılır. Burada 0 yerine ne yazılırsa, dizinin o elemanını elde edilir. Diyelim ki, 5. elemanın (yani `dizi[4]`) kullanılması isteniyor, o zaman `*(ptr + 4)` yazılmalıdır.

4. İşaretçiler ve Diziler

```
1  #include<stdio.h>
2  #include<locale.h>
3  int main( void )
4  {
5      setlocale(LC_ALL, "Turkish");
6      int i;
7      int dizi[ 5 ] = { 5,15,10,14,21};
8      int *ptr;
9      ptr = dizi;
10     printf( "\n Elemanın Değeri      Adresi      Adresi\n");
11     printf( "\n _____\n");
12     for( i = 0; i < 5; i++ )
13     {
14         printf( " %d          %p          %p \n",
15                *( ptr + i ), &( dizi[i] ), ptr+i );
16     }
17     return 0;
18 }
```

Program 5.8

Elemanın Değeri	Adresi	Adresi
5	000000000061FE20	000000000061FE20
15	000000000061FE24	000000000061FE24
10	000000000061FE28	000000000061FE28
14	000000000061FE2C	000000000061FE2C
21	000000000061FE30	000000000061FE30

Program 5.8 'in Ekran Çıktısı

4. İşaretçiler ve Diziler

```
1  #include<stdio.h>
2  #include<locale.h>
3  int main( void )
4  {
5      setlocale(LC_ALL, "Turkish");
6      int i;
7      int dizi[ 5 ] = { 5,15,10,14,21};
8      int *ptr;
9      ptr = dizi;
10     printf( "\n Elemanın Değeri      Adresi      Adresi\n");
11     printf( "\n _____\n");
12     for( i = 0; i < 5; i++ )
13     {
14         printf( " %d          %p          %p \n",
15                 *( ptr + i ), &( dizi[i] ), ptr+i );
16     }
17     return 0;
```

Program 5.8

Elemanın Değeri	Adresi	Adresi
5	000000000061FE20	000000000061FE20
15	000000000061FE24	000000000061FE24
10	000000000061FE28	000000000061FE28
14	000000000061FE2C	000000000061FE2C
21	000000000061FE30	000000000061FE30

Program 5.8 'in Ekran Çıktısı

Kaynaklar

Veri Yapıları ve Algoritmalar – Dr. Rifat ÇÖLKESEN,
Papatya yayıncılık

Veri Yapıları ve Algoritmalar-Dr. Öğ. Üyesi Ömer
ÇETİN

Veri Yapıları – Prof. Dr. Erkan TANYILDIZI