BÖLÜM 8: DİZİLER

Dizi bir kümedir. Aynı tipte verilere tek bir isimle erişmek için kullanılır. Bir dizinin bütün elemanları bellekte peş peşe saklanır.

Dizi Tipi	Genel formatı
Tek boyutlu	`tip degisken_adi[eleman_sayısı];`
	örnek: `int a[10];`
İki boyutlu	`tip degisken_adi[satir_say][sütun_say];`
	örnek: `float b[3][4];`
Çok boyutlu	`tip degisken_adi[boyut_1][boyut_2]...[boyut_n];`
	örnek: `double x[2][4][2];`

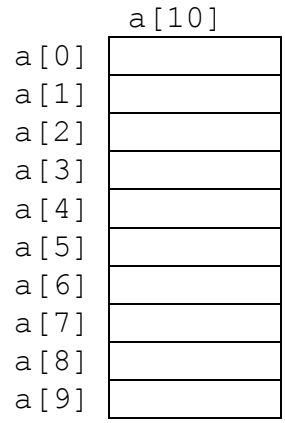
C programlama dilinde, diğer programlama dillerinde olduğu gibi dizi değişken tanımlamak yukarıdaki gibi mümkündür. C programlama dilinde ilk indis numarası 0 ile başlar.

I. DİZİ DEĞİŞKENLER

Dizi değişkenler, daha önce görmüş olduğumuz bütün değişken tiplerinde tanımlanabilir. Dizi değişken tanımlanırken, önce değişkenin hangi tip olduğu, daha sonra da değişken adı yazılır. Değişken adının yanına da köşeli parantez içersinde eleman sayısı yazılır.

```
int a[10];
```

Böyle bir tanımdan sonra `a` dizi değişkenine, bilgisayar belleğinde ayrılan alan aşağıdaki gibidir.



Yukarıdaki şekilde de görüldüğü gibi, `int a[10]` tanımlaması sonunda, bellekte 10 bellek hücresi ayrılmaktadır. C programlama dilinde ilk indis numarası 0'dır. `int a[10]` tanımlaması sonucunda, bellekte her biri bir integer tipi sayıyı tutabilecek 10 adet bellek hücresi ayrılmıştır. Bu bellek hücrelerinden herhangi birisine değişken adı ve yanında köşeli parantez içersinde indis numarası ile ulaşılır. İndis sayısı, tamsayı bir sabit olmalıdır.

Dizi değişkenler, yukarıdaki örnekteki gibi tek boyutlu olarak tanımlanabileceği gibi iki ve daha fazla boyutlu olarak da tanımlanabilirler.

```
int b[3][4];
```

Bu tanımlama sonucunda, bilgisayarın belleğinde ayrılan hücreler aşağıdaki gibidir.

b[3][4]

b[0][0]	b[0][1]	b[0][2]	b[0][3]
b[1][0]	b[1][1]	b[1][2]	b[1][3]
b[2][0]	b[2][1]	b[2][2]	b[2][3]

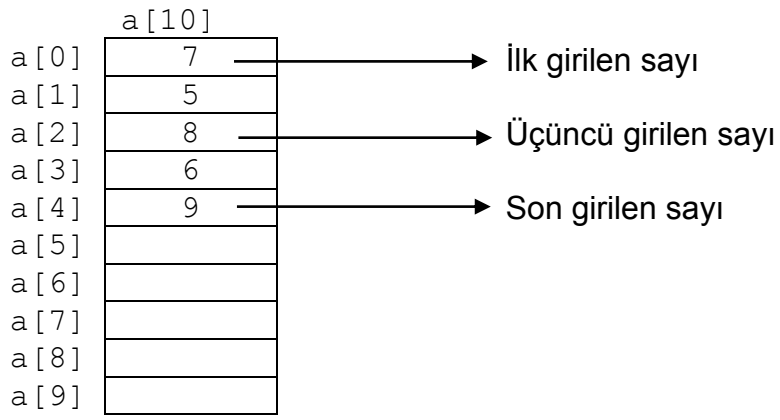
Değişken adının sağında yer alan köşeli parantezlerden birincisinin içindeki rakam satır sayısını, ikincisinin içindeki rakam ise sütun sayısını vermektedir. Yukarıdaki bellek hücrelerinden birisine ulaşmak için, değişken adının yanına köşeli parantezler içinde satır ve sütun numaralarını yazmak gerekir. Örneğin `b[0][3]` yazılarak sağ üst köşedeki bellek hücresine giriş-çıkış yapılabilir. Aşağıdaki programda dizi değişken kullanılmıştır.

```
#include <stdio.h>

void main ()
{   int a[10];
    int i;

    for(i=0;i<5;i++)
    {   printf("%d. sayiyi giriniz :",i+1) ;
        scanf ("%d", &a[i]);
    }

    printf("\n") ;
    printf("Ilk girdiginiz sayi: %d\n",a[0]);
    printf("Ucuncu girdiginiz sayi: %d\n",a[2]);
    printf("Son girdiginiz sayi: %d\n",a[4]);
}
```

Program çıktısı:

```
1.  sayıyı giriniz: 7
2.  sayıyı giriniz: 5
3.  sayıyı giriniz: 8
4.  sayıyı giriniz: 6
5.  sayıyı giriniz: 9

İlk girdiğiniz sayı: 7
Üçüncü girdiğiniz sayı: 8
Son girdiğiniz sayı: 9
```

Program çalıştırıldığında 10 hücreli bir bellek alanı ayrılır ve `for` döngüsü vasıtasıyla ilk 5 hücreye 5 adet sayı girilir. Tanımlanmış ve bellekte ayrılmış olan bütün hücrelerin kullanılması zorunlu değildir. Program çalıştırıldıktan sonra bellekte ayrılan hücreler ve içlerindeki sayılar aşağıdaki şekilde görülmektedir.



Aşağıdaki programda, klavyeden girilen 5 sayı, `for` döngüsü kullanılarak görüntülenmektedir. Programda integer tipi dizi değişken kullanılmıştır. `i` değişkeni ise `for` döngülerini kontrol eden değişkendir.

```
#include <stdio.h>
void main()
{   int i, a[10];
    for(i=0;i<5;i++)
    {   printf("%d. sayıyı giriniz :",i+1);
        scanf ("%d", &a[i]);
    }
    printf("\n");
    for (i=0;i<5;i++)
        printf("%d. sayı :%d\n",i+1,a[i]);
}
```

Yukarıdaki program çalıştırıldığında ilk `for` döngüsü, klavyeden girilen 5 adet sayıyı `a` dizi değişkeninin ilk 5 hücresine aktarır. İkinci for döngüsü ise ilk 5 bellek hücresinde bulunan sayıları görüntülemektedir.

Şimdi inceleyeceğimiz programda ise, klavyeden girilen 10 adet sayının en büyük ve en küçük olanı, sayıların toplamı ve ortalaması bulunup görüntülenmektedir.

```
#include <stdio.h>
void main()
{   int i, min, max, toplam, a[10];
    float ortalama;
    for(i=0;i<10;i++)
    {   printf("%d. sayıyı giriniz :",i+1);
        scanf ("%d", &a[i]);
    }
    min=max=a[0];
    toplam = a[0];
    for (i=1;i<10; i++)
    {   if(a[i]>max)
            max=a[i];
        if(a[i]<min)
            min=a[i];
        toplam = toplam + a[i];
    }
```

```
    ortalama = (float)toplam / 10; printf("\n");
    printf("Sayilarin en buyugu :%5d\n", max);
    printf("Sayilarin en kucugu :%5d\n", min);
    printf("Sayilarin toplami   :%5d\n", toplam);
    printf("Sayilarin ortalamasi:%8.2f\n", ortalama);
}
```

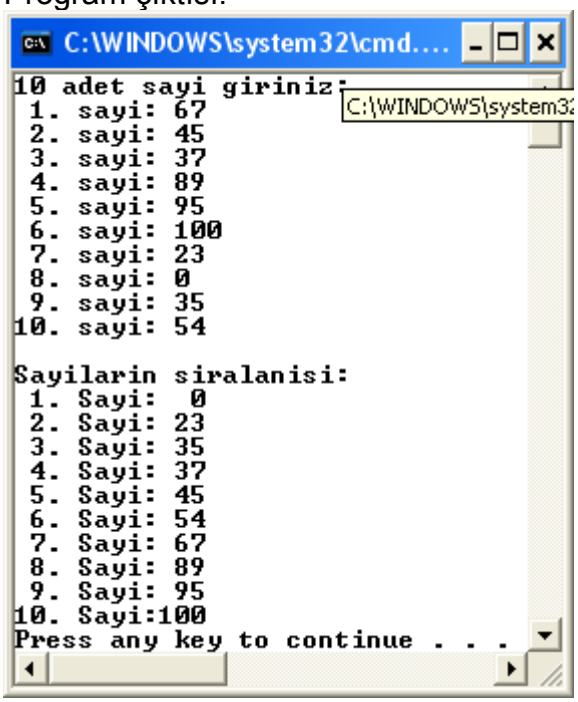
Bu program çalıştırıldığında, ilk `for` döngüsüyle, klavyeden 10 adet tam sayı girilir ve `a` dizi değişkeninin 10 adet hücresine aktarılır. Daha sonra `a[0]` hücresinde bulunan sayı `max` ve `min` değişkenlerine aktarılır. Böylece ilk sayıyı geçici bir süre en büyük ve en küçük sayı kabul ederiz. Daha sonra `max` ve `min` değişkenlerinin içindeki sayılar, sırayla `a[1], a[2], a[3] ...`ve `a[9]` hücrelerinde bulunan sayılarla karşılaştırılır. Bu ikili karşılaştırmalarda varsa daha büyük olan sayı `max` değişkenine, daha küçük olan sayı ise `min` değişkenine aktarılır. `toplam` değişkenine ise başlangıç değeri olarak dizinin ilk elemanının, yani `a[0]` değeri aktarılıp, geri kalan dizi elemanları ise `for` döngüsü içersinde `toplam` değişkenine eklenmektedir. Daha sonra `ortalama` değeri de hesaplanıp `printf` fonksiyonu kullanılarak `max`, `min`, `toplam` ve hesaplanan `ortalama` değerleri ekrana yazdırılmaktadır.

Bilgisayarların bize sağladığı en önemli kolaylıklardan bir tanesi de sayıları ve kelimeleri çok hızlı olarak işleyip sıralayabilmesidir. Sayılar büyükten küçüğe ya da küçükten büyüğe, kelimeler ise ‘a’dan ‘z’ ye veya ‘z’ den ‘a’ ya doğru sıralanabilir.

Programcılık tekniği açısından çeşitli sıralama metotları bulunmaktadır. Bunlar içinde en yaygın olarak kullanılanlardan bir tanesi BUBBLE SORT (köpük sıralama) metodudur. BUBBLE SORT metodunda, sıralanacak sayıların bulunduğu dizi değişken hücrelerine ulaşmak üzere iç içe kurulmuş iki döngü kullanılır. Aşağıdaki programda klavyeden girilen *N* adet sayı büyükten küçüğe doğru sıralanmakta ve görüntülenmektedir.

```
#include <stdio.h>
#define N 10
void main()
{   int a[N], i, j, gec;
    printf("%d adet sayi giriniz:\n", N);
    for(i=0; i<N; i++)
    {   printf("%2d. sayi: ",i+1);
        scanf("%d", &a[i]);
    }
    for(i=1; i<N; i++)
        for(j=0; j<N-1; j++)
            if(a[j] > a[j+1])
            {   gec = a[j];
                a[j] = a[j+1];
                a[j+1] = gec;
            }
    printf("\nSayilarin siralanisi:\n");
    for(i=0; i<N; i++)
        printf("%2d. Sayı:%3d\n",i+1, a[i]);
}
```

Program çıktısı:



Sıralanacak olan sayılar, birinci `for` döngüsü ile dizi değişkenin 10 elemanına aktarılır. Daha sonra iç içe iki döngü yardımıyla `a[0]` hücresinden başlamak üzere sayılar sırayla bir sonraki sayılarla tek tek karşılaştırılır. Eğer ilk hücrede bulunan sayı, sonraki hücrede bulunan sayıdan büyükse, bu iki sayı yer değiştirir. `gec` isimli değişken, iki bellek hücresinde bulunan sayıların kaybolmadan yer değiştirmesi amacıyla kullanılan, bilginin geçici olarak saklandığı bir değişkendir. Bu işlem en büyük sayıyı dizinin sonuna atacak olup bu işlem 9 kez tekrarlanır.

Şimdi de iki boyutlu bir dizi değişkenin kullanıldığı, 1 ile 10 arasındaki sayılardan herhangi birisinin karesini görüntüleyen bir programı inceeyelim.

```
#include <stdio.h>
void main()
{   int i,j;
    int kare[10][2] = {1, 1, 2, 4, 3, 9, 4, 16, 5, 25,
                       6, 36, 7, 49, 8, 64, 9, 81, 10, 100};
    printf("1 ile 10 arasinda bir sayi giriniz:");
    scanf("%d",&i);
    for(j=0;j<10; j++)
        if (kare[j][0]==i)
            printf("%d nin karesi %d dir.\n",i,kare[j][1]);
}
```

Program çıktısıı:

```
1 ile 10 arasinda bir sayı giriniz: 6
6 nin karesi 36 dır.
```

Bu programda kare dizi değişkenine değer atama işlemi değişkenin tanımlandığı yerde yapılmıştır. Bu değer atama işlemi sonunda, `kare[10][2]` dizi değişkeninin elemanlarındaki değerler aşağıdaki tabloda gösterilmiştir.

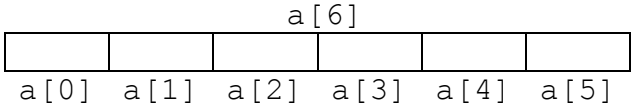
	0	1
0	1	1
1	2	4
2	3	9
3	4	16
4	5	25
5	6	36
6	7	49
7	8	64
8	9	81
9	10	100

Programdaki `for` döngüsü, girilen sayı dizi değişkenin 1. sütunundaki (yani 0 yazan sütun) sayılardan birisine eşit olana kadar devam eder. Girilen sayı ile 1. sütundaki hücrelerden birisindeki sayı eşit olduğunda, `break` komutu çalışır ve döngü dışına çıkılır. O andaki `j` değeri 2. sütunda bulunan ve girilen sayının karesini tutan sayının bulunduğu bellek elemanının satır numarasıdır.

II. `char` TİPİ DİZİLER

Daha önce, değişken tipleri incelenirken, C programlama dilinde string diye bir değişken tipi bulunmadığını, bunun yerine `char` tipi değişkenlerin dizi olarak tanımlandıktan sonra string ifadeleri tuttuklarını belirtmiştik.

Örneğin `char a[6]` şeklindeki bir tanımlamadan sonra, bellekte aşağıdaki şekilde görüldüğü gibi bir bellek bölgesi `a` değişkeni için ayrılır.



Bu bellek elemanlarından her biri bir karakter tutabilir. String ifadelerin son elemanından bir sonraki hücreye, derleyici tarafından string sonu karakteri (`\0`) konulacağı için, örneğin 5 harflik bir kelimeyi tutmak üzere en az 6 elemanlı bir `char` tipi dizi değişken tanımlanmalıdır.

Birden fazla string tipi sabitin bir dizi değişkenine atanması gerektiğinde iki boyutlu `char` tipi bir dizi değişken tanımlanması gerekir.

```
char isim[20][30];
```

Bu tanımlamada `isim` değişkeninin sağ tarafındaki 20 sayısı, isim sayısını, 30 sayısı ise her ismin kaç karakterden oluşabileceğini gösteriyor. Ancak 30 bellek elemanından bir tanesinin string sonu (`\0`) ifadesi için kullanılacağını biliyoruz. Bu nedenle yukarıda tanımlanan iki boyutlu, `char` tipi dizi değişkenine uzunlukları en fazla 29 karaktere kadar olan 20 isim yerleştirilebilir.

Dizilerin gerçek gücü, yukarıdaki örneklerde de görüldüğü gibi, döngülerle kullanıldığında görülmektedir. Örneğin bir okul için öğrenci takip programı yazdığımızı düşünelim ve öğrenci adedinin ise 500 olduğunu varsayalım. Şimdilik

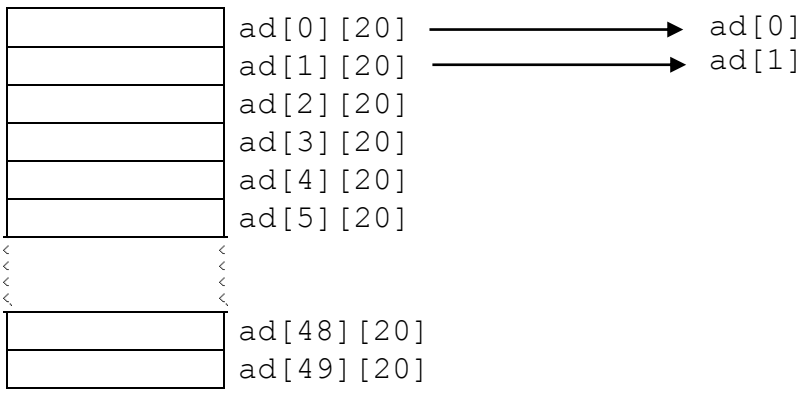
sadece bu öğrencilerin adlarının tutulacağı bir program yazmamız gerektiğinde, diziler olmadan, her bir öğrenci için bir değişken tanımlaması yapıp kullanmamız gerekecekti. Dizileri kullanarak 500 adet öğrenciyi tek bir dizi değişken bünyesinde tutmamız mümkündür.

Aşağıdaki programda, klavyeden 5 isim girilmekte ve daha sonra bu isimler görüntülenmektedir.

```
#include <stdio.h>
#include <conio.h>
void main()
{   char ad[50][20];
    int i;
    for(i=0; i<5; i++)
    {
        printf("%d. kişinin adını giriniz:",i+1);
        scanf ("%s", ad[i]);
    }
    printf ("\n");
    printf("Liste için herhangi bir tuşa basınız.. \n") ;
    getch();
    printf("\n");
    for (i=0;i<5; i++)
        printf("%s\n",ad[i]);
}
```

Bu programda, `char ad[50][20]` tanımlamasından sonra bellekte, her biri 19 karakter genişliğinde (çünkü bir hücreye string sonu karakteri "`\0`" yazılacaktır) 50 kelime alacak şekilde yer ayrılır.

Dikkat edilirse `for` döngüsünün tekrarlama bloğunda, `ad` dizi değişkeni `scanf` fonksiyonunun parametresi olarak kullanılırken `ad[i]` şeklinde gönderme yapılmıştır. Yani `ad` değişken isminden sonra gelen iki köşeli parantezden sadece birincisi kullanılmıştır. Buna göre, kelimenin karakter olarak uzunluğunu veren sayının bulunduğu köşeli parantezin kullanılmasına gerek olmaksızın, dizi değişkenin herhangi bir elemanına gönderme yapılabilir. Ancak, değişkenlerin tanımlandığı bölümde, dizi değişkenin hem kaç string ifadeyi tutacağı ve hem de her stringin kaç karakterden oluşacağı belirtilir. Örneğin `char ad[50][20]` tanımlaması sonucunda bellekte aşağıdaki düzenleme oluşur.



Yukarıdan aşağıya doğru bütün bellek hücreleri 20 karakter tutacak genişliktedir. İkinci köşeli parantez içersindeki 20 rakamı hiç değişmediğine göre, herhangi bir hücreye gönderme yapılırken sadece ilk köşeli parantezin içersindeki sayının verilmesi yeterlidir. Örneğin `ad[2][20]` hücresine gönderme yapılırken `ad[2]` yazılması yeterlidir.

Bilgisayarların bize sağladığı en büyük kolaylıklardan bir tanesi de, daha önceden girilmiş olan bilgilerin çok kısa sürede çağrılabilmesidir. Bilgisayara çeşitli bilgiler girilir ve bilgi havuzunda toplanır. Daha sonra, deyim yerindeyse çeşitli filtreler kullanılarak belirli özellikleri taşıyan bilgiler bulunur ve kullanılır. Aşağıdaki programda önce klavyeden isimler ve numaralar girilmekte, daha sonra da ismi verilen kişinin numarası bulunarak görüntülenmektedir.

```
/* İsme göre arama yapan program */
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main()
{   char ad[50][20];
    char ara[20];
    int i, bulundu, no[50];
    char cevap = 'e';

    for(i=0;i<5;i++)/*isim ve numaraların girildiği blok */
    {   printf("%d. kisinin adini giriniz:",i+1);
        scanf("%s", ad[i]);
        printf("%d. kisinin numarasini giriniz:",i+1);
        scanf ("%d", &no[i]);
        printf("\n");
    }

    while(cevap == 'e')
    {   bulundu=0;
        printf("Aradiginiz kisinin adini giriniz:");
        scanf("%s", ara);
        printf("\n");
        for(i=0;i<5;i++)    /* ismin arandığı blok */
            if (strcmp(ara, ad[i])==0)
            {   bulundu=1;
                printf("Adi: %s\nNumarasi: %d\n",ad[i], no[i]);
            }
        if(bulundu==0)
            printf("Aradiginiz isim bulunamadi..\n\n");
        printf("Arama islemine devam edecek misiniz (e/h):");
        cevap = getche();
        printf("\n");
    }
}
```

Bu program çalıştırıldığında klavyeden 5 isim ve bu isimlere ait numaralar girilir. İsimler `ad[50][20]` dizi değişkenine, numaralar ise `no[50]` dizi değişkenine aktarılmaktadır. 5 kişiye ait isim ve numaralar girildiğinde bellekteki yerleşim düzeni aşağıdaki şekilde gösterilebilir.

ad[0]	ali
ad[1]	burak
ad[2]	osman
ad[3]	kadir
ad[4]	ege
ad[5]	nihal

no[0]	55
no[1]	67
no[2]	20
no[3]	45
no[4]	32
no[5]	76

ad[48]	
ad[49]	

no[48]	
no[49]	

Yukarıda da görüldüğü gibi ilk girilen isim `ad[0]` değişkenine, ilk girilen numara da (yani ilk girilen isme ait numara) `no[0]` değişkenine aktarılmıştır. Dolayısıyla isimlerle, o isimlere ait numaralar indis numaraları aynı olan değişkenlerde tutulmaktadır. Örneğin `ad[3]` değişkeninde ismi yazılı olan kişinin numarası `no[3]` değişkenindedir.

Bu programda kullanılan `strcmp(s1, s2)` fonksiyonu, parametre olarak verilen `s1` ve `s2` string değerlerini karşılaştırır. Eğer bu iki string birbirine eşit ise 0 sayısı, `s1 s2`'den büyükse 1 ve `s1 s2`'den küçük ise -1 değerini üretir.

Programın arama yapan bölümünde, `ara[]` değişkenine girilmiş olan isim, bir `for` döngüsü yardımıyla `ad[0]`'dan başlamak üzere 5 adet dizi elemanındaki isimlerle karşılaştırılır, `ara[]` değişkeninin tuttuğu isimle bellek hücrelerinden herhangi birisinde bulunan bir isim eşit olduğunda `strcmp()` komutu 0 değerini üreteceği için i`f(strcmp(ara, ad[i])==0)` koşulu gerçekleşmiş olur ve aranılan ad ve numara görüntülenir. .

Aşağıdaki programda klavyeden 5 kişinin ismi ve numarası girilmekte, daha sonra da numarası verilen kişinin ismi görüntülenmektedir.

```
/* Numaraya göre arama yapan program */
#include <stdio.h>
#include <conio.h>

void main()
{   char ad[50][20];
    int ara;
    int i, bulundu, no[50];
    char cevap = 'e';

    for(i=0;i<5;i++)/*isim ve numaraların girildiği blok */
    {   printf("%d. ogrencinin adini giriniz:",i+1);
        scanf("%s", ad[i]);
        printf("%d. ogrencinin numarasini giriniz:",i+1);
        scanf ("%d", &no[i]);
        printf("\n");
    }
```

```
    while(cevap == 'e')
    {   bulundu=0;
        printf("Aradiginiz ogrencinin numarasini giriniz:");
        scanf("%d",&ara);
        printf("\n");
        for(i=0;i<5;i++)    /* numaranın arandığı blok */
            if (ara == no[i])
            {   bulundu=1;
                printf("Numarasi: %d\nAdi: %s \n", no[i], ad[i]);
            }
        if(bulundu==0)
            printf("Aradiginiz numara bulunamadi..\n\n");
        printf("Arama islemine devam edecek misiniz (e/h):");
        cevap = getche();
        printf("\n\n");
    }
}
```

Bu programda da aranılan öğrencinin numarası `ara` değişkenine girilmekte, bu numara `no` dizi değişkenine girilmiş olan numaralarla tek tek karşılaştırılmaktadır.

Dizi değişkenlerin kullanımı ile ilgili örnekler. Örnekleri dikkatle inceleyiniz.

III. GENEL ÖRNEKLER

Örnek 1: Dizi elamanlarına erişim

```
#include <stdio.h>
void main()
{   int i, x[5];
    x[0]=5;
    x[1]=25;
    x[2]=-40;

    for(i=0;i<3;i++)
       printf("%d\n",x[i]);
}
```

Örnek 2: Dizilere başlangıç değeri verme

```
#include <stdio.h>
void main()
{   int i, x[5]={5, 0, 6};
    int y[]={1, 5, 9};

    puts("x\ty");
    for(i=0;i<3;i++)
       printf("%d\t%d\n",x[i],y[i]);
}
```

Örnek 3: Karakter dizisi (string)

```
#include <stdio.h>
void main()
{   int i;
    char ad [][15]  = {"Ahmet", "Mehmet", "Can"};
    int kilo[]={70, 60, 52};
    float boy[] ={1.70, 1.85, 1.45};
    puts("\nISIM\tKILO\tBOY");
    for(i=0; i<3; i++)
        printf("%s\t%d\t%4.2f\n", ad[i], kilo[i], boy[i]);
}
```

Örnek 4: Beş sayının ortalaması

```
#include <stdio.h>
void main()
{   int i, x[5], toplam = 0;
    float ort;

    for(i=0; i<5; i++)
    {
        printf("%d. eleman : ", i+1);
        scanf("%d", &x[i]);
        toplam += x[i];
    }
    ort = (float)toplam/5;
    printf("Ortalama: %f", ort);
}
```

Örnek 5: Dizinin bellekte kapladığı alan

```
#include <stdio.h>
 void main()
{   char dizi1[10];
    int dizi2[10];
    float dizi3[10];
    double dizi4[10];
    printf("%d\n",sizeof(dizi1));
    printf("%d\n",sizeof(dizi2));
    printf("%d\n",sizeof(dizi3));
    printf("%d\n",sizeof(dizi4));
}
```

Örnek 6: İki matrisin toplamı

```
#include <stdio.h>
void main()
{
    int i, j, A[3][3], B[3][3], C[3][3];
    puts("iki matrisin toplami:");
    puts("A matrisinin elemanlarini giriniz:");
```

```
    for(i=0;i<3;i++)
    {   for(j=0;j<3;j++)
        {   printf("A(%d,%d)=",i+1,j+1);
             scanf("%d",&A[i][j]);
        }
    }
    puts("B matrisinin elemanlarını giriniz:");
    for(i=0;i<3;i++)
    {   for(j=0;j<3;j++)
        {   printf("B(%d,%d)=",i+1,j+1);
            scanf("%d",&B[i][j]);
        }
    }
    puts("A+B matrisinin elemanları:");
    for(i=0;i<3;i++)
    {   for(j=0;j<3;j++)
        {   C[i][j] = A[i][j] + B[i][j];
            printf("C(%d,%d)=%d\n",i+1,j+1,C[i][j]);
        }
    }
}
```

Örnek 7: Dizi elemanlarının bellekteki sıralanış adresleri

```
#include <stdio.h>
void main()
{   int i, x[10];
    puts("x dizisinin elemanlarinin bellekteki sıralanışı");
    for(i=0;i<10;i++)
        printf("%p\n",&x[i]);
}
```