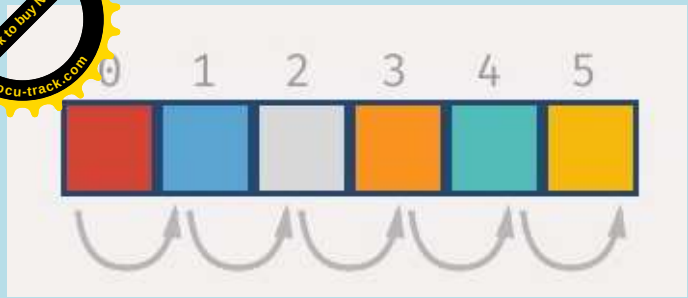
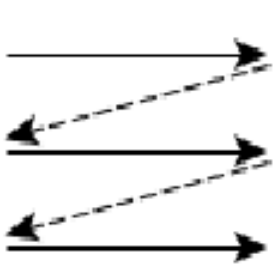
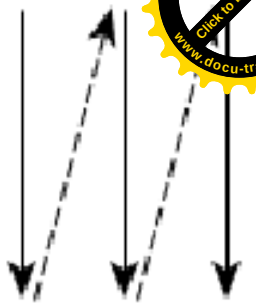


Diziler ve Dizi işlemleri

0	8	4
5	3	5
2	3	6





0	8	4	5	3	5	2	3	6

0	5	2	8	3	3	4	5	6

40	55	63	17	22	68	89	97	89
0	1	2	3	4	5	6	7	8

<- Array Indices

Array Length = 9

First Index = 0

Last Index = 8

Diziler

- C dilinde aynı tipteki verilere tek bir isimle erişebilmek için diziler kullanılır.
- Diziler aynı tipteki veriler kümesidir.
- Diziler bir veya çok boyutlu olabilir.
- C dilinde diziler ile çalışmak esneklik sağlar. Kullanımına göre statik veya dinamik yapıdaki diziler tanımlanabilir.
- Eğer dizinin kaç elemanlı olacağı başlangıçta belirli ise, program başında dizi bildirimi yapılır ve derleyici program süresince kullanılmak üzere gerekli bellek alanını ayırır.
- Dinamik bildirimde ise dizi için gerekli bellek alanı programın yürütülmesi esnasında sistem bellek yöneticisinden istenir ve kullanılır.

PDF-XChange Viewer Click to buy NOW! www.docu-track.com

Dizi kullanmanın Önemi

- 1000 adet adı, soyadı ve notu bilgisini saklamak için;
- Adı…..1000Adet,
- Soyadı…..1000Adet
- Notu…..1000Adet
- Toplam **3000** adet değişkenin kullanılması gereklidir.

Dizilerin Bildirimi

- Bir dizi çok sayıda değişken barındırdığından, bunları birbirinden ayırt etmek için ***indis*** adı verilen bir bilgiye ihtiyaç vardır. C Programlama Dili'nde, bir dizi hangi tipte tanımlanmış olursa olsun başlangıç indisi her zaman 0'dır.

- Bir dizinin bildirim işleminin genel biçimi söyledir: **veriTipi** ***dizi_adı[eleman_sayısı];***
- Örneğin, 5 elemanlı, kütle verilerini bellekte tutmak için, kutle dizisi şöyle tanımlanabilir:
- **float kutle[5];**

İndisli değişkenler

- Dizi içerisinde her elemanın bir indis numarası bulunur. Bu nedenle "indisli değişkenler" olarak da adlandırılırlar.
- İndis Numarası her elemanın dizi içerisindeki yerini, yani kaçıncı elemanı olduğunu gösterir.
- Bir dizi şu şekilde tanımlanır;
- <Veri Türü> <Dizi_İsmi> [Eleman Sayısı]
  - **int ders_notu[10];**
- Dizi değişken tanımlamak için kullanılan tanım cümleleri diğer değişkenler için kullanılan cümlelerden farksızdır. Tek farklılık, tanımlanan değişkenin dizi değişken olduğunu gösteren ve değişken isminin hemen yanında bulunan [ ] (köşeli parantez) dir.

- Bildirim sırasında dizilerin eleman sayısı tamsayı türünden bir sabit ifadesiyle belirtilmesi zorunludur. Örneğin:
- int n = 100;
- int a[n];
- şeklindeki tanımlama, dizi uzunluğunun değişken (n) ile belirtilmesi nedeniyle geçersizdir.
- Bunun yerine, dizilerin eleman sayısı aşağıdaki gibi sembolik sabitlerle belirtmek mümkündür.
- #define n 100
- ...
- int a[n];

Bir Boyutlu Diziler

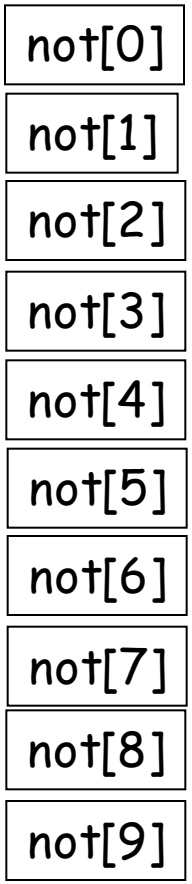
Dizi bildirimi;

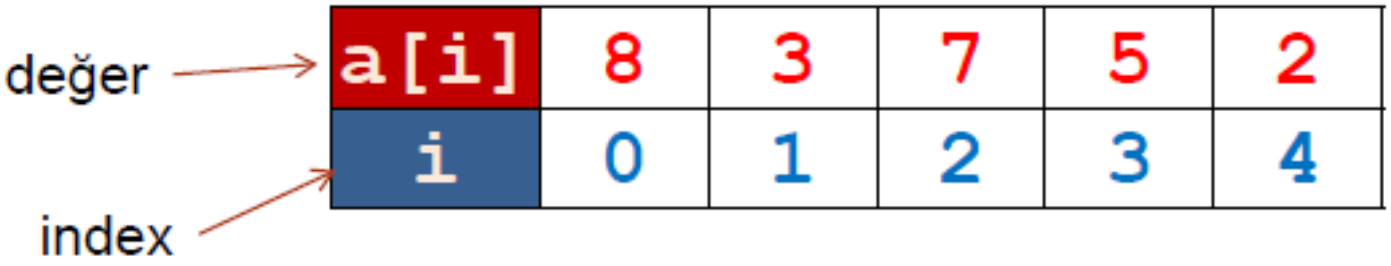
tip ad [eleman sayısı];

olarak yapılır.

int not[10];

gibi bir bildirimde bellekte 10 tane tamsayının saklanacağı alan ayrılır ve bu alanın başlangıç adresini dizi adına atar. Dizinin ilk indisi "0" dır.






```
i=5;
int a[i] = {8,3,7,5,2};
```

veya

```
a[0]=8;
a[1]=3;
a[2]=7;
a[3]=5;
a[4]=2;
```

- Dizinin bellekte işgal ettiği alan, dizi boyutu ile elemanların tip uzunluğunun (sekizli) çarpımı ile bulunabilir. Yukarıdaki dizi bellekte;
- 5*2 sekizli=10 sekizli yer tutar.
- Bir dizinin bellekte kapladığı alanın bayt cinsinden karşılığı **sizeof** operatörü ile öğrenilebilir.
- int a[5], b, c; ... b = sizeof(a);
- /* bellekte kapladığı alan: b = 4*5 = 20 bayt */

Dizilere Başlangıç Değeri Verme


```
float kutle[5]= { 8.471, 3.683, 9.107, 4.739, 3.918 };
int maliyet[3] = { 25, 72, 94 };
double a[4] = { 10.0, 5.2, 7.5, 0.0};
```

Bir dizinin uzunluğu belirtilmeden de başlangıç değeri atamak mümkündür.

```
int a[ ] = { 10, 20, 30, 40 };
float v[ ] = { 9.8, 11.1};
```

Dizilerle Yazma ve Okuma;

```
not[0]=2376;
not[5]=6;
t= not[0];
scanf("%d",&not[7]);
printf("%d",not[7]);
```

şeklinde kullanılır.

Ekrandan dizi okutma ve yazdırma


```
#include <stdio.h>
int main() {
  int i, dizi[5];
  printf("dizi elemanlarini gir= \n");
  for(i=0; i<4; i++)
     scanf("%d",&dizi[i]);
  printf("dizi elemanlari:\n");
  for(i=0; i<4; i++)
     printf("%d\n", dizi[i]);
  getch(); }
```

```
#include <stdio.h>
int main() {
  int i;
  int dizi[5]= {5,16,7,19,44};
  for(i=0; i<=4; i++)
     printf("%d. eleman :%d\n",
i+1,dizi[i]);
 getch(); }
```

Örnek: klavyeden girilen $N = 10$ adet sayının ortalamasını diz kullanarak yapan program.


```
#include <stdio.h>
#define N 10
int main() {
  int i;
  float x[N], ort, toplam;
  for(i=0; i<N; i++) {
      printf("%d. sayi : ", i+1);
      scanf("%f",&x[i]);
      toplam += x[i]; }
      printf("Sayilarin toplami= %f\n", toplam);
  ort = (float)toplam/N;
  printf("Sayilarin ortalamasi = %f\n", ort);
 getch(); }
```

Örnek: diziye rasgele sayılar atamak ve ekrana yazdırma

PDF-XChange Viewer Click to buy NOW! www.docu-track.com


```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
main()  {

int not[10],k;
    for(k=0;k<10;k++)  {
        not[k]=rand()%100;
        printf("not[%d]=%d\n",k, not[k]);   }
getch();
}
```

Örnek: Bir dizide maksimum ve minimum değerlerin bulunması


```
#include<stdio.h>
int main( void )  {
    int dizi[ ] = { 15, 54, 1, 44, 55, 40, 60, 4, 77, 45 };
    int i, max, min;
    min = dizi[ 0 ]; //dizideki ilk elemanı min kabul ediyoruz
    max = dizi[ 0 ]; //dizideki ilk elemanı max kabul ediyoruz
     for( i = 1; i < 10; i++ ) {
         if( min > dizi[i] )
         min = dizi[i];
         if( max < dizi[i] )
         max = dizi[i];
       }
printf( "Dizinin en kucugu: %d\n", min );
printf( "Dizinin en buyugu: %d\n", max );
  getch();   }
```

Dizilerin Fonksiyonlara Aktarılması

C programlamada bir dizi yerel değişken olarak bildirilmişse ve bir fonksiyonda kullanılacaksa, dizinin adı fonksiyon parametresi olarak kullanılır. Dizinin adı yazılarak aslında dizinin bellekte işgal ettiği alanın ilk adresi fonksiyona gönderilmektedir.

Fonksiyonlara parametre aktarımı bilindiği gibi yığın üzerinden yapılır. Böylece tüm dizi elemanlarını yığına atmak yerine tek bir veri, dizi başlangıç adresini ifade eden dizi adını kullanmak mantıklıdır.

Dizi kaç boyutlu olursa olsun, sadece adını fonksiyona aktarmak yeterlidir. Gönderilen dizinin boyut bilgisi, o fonksiyon tanımlanırken formal değişken bildirim kısmında verilir.

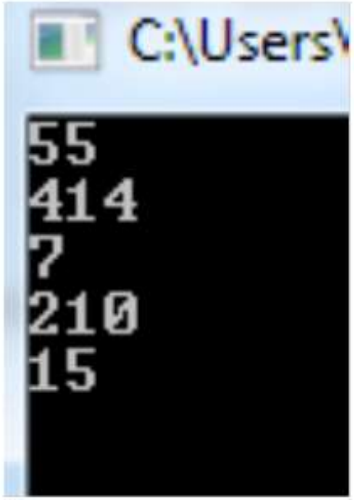
```
main() {
float kur [2][3]={3.2, 5.4, 6.5, 1.0, 0.5, 6.5};
……
k=ortalama(kur);
……}
ortalama (float gelen[2][3])        // Dizi 1 boyutlu ise ortalama (float gelen[])
{……}
```

Örnek 1: Dizilerin fonksiyonlara aktarımı


```
#include<stdio.h>
void elemanlari_goster( int [ 5 ] );

int main( void ) {
int dizi[ 5 ] = { 55, 414, 7, 210, 15 };
goster( dizi );
getch(); }

void goster( int goster_dizi[5] ) {
   int i;
   for( i = 0; i < 5; i++)
      printf( "%d\n", goster_dizi[ i ] );   }
```



Örnek2: Dizilerin Fonksiyonlara Aktarılması

/*baslangic degerleri verilen dizinin en küçük elemanını bulur ve ekrana yazar. bul adli fonksiyon kendine gelen dizinin en kucuk elemanı bulur ve cagiran fonksiyona gönderir. */

```
#include<stdio.h>
#include<conio.h>
#define n 10

bul (int dizi[])     {
   int i,ek;
   ek=dizi[0];
   for(i=0;i<n;i++)
      if(dizi[i]<ek)
           ek=dizi[i];
   return ek;        }
```

```
main()  {
 int not[n]={23,56,9,78,96,55,34,21,7,8};
 int enkucuk;

 enkucuk=bul(not);
 printf("dizideki en kucuk eleman %d
dir", enkucuk);
getch();
}
```

Dizi Uygulaması: Varyans ve Standart Sapma


```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
main()  {
  int k,n=10;
  float toplam=0.0;
  float ort,varyans,ssd;
  float
a[10]={2.1,5.4,7.6,6.0,6.5,9.8,5.0,3.4,2.5,9.0};
   for(k=0;k<n;k++)
      ort+=a[k];
  ort/=n;
  for(k=0;k<n;k++)
    toplam+=(a[k]-ort)*(a[k]-ort);
  varyans=toplam/n;
  printf("varyans=%f\n",varyans);
  ssd=sqrt(varyans*n/(n-1));
  printf("ssapma=%f",ssd);
getch(); }
```

$$V = \frac{1}{N}\sum_{i=1}^{N}(D_i - Ort)^2$$

$$ssd = \sqrt{\frac{V * N}{N-1}}$$

Burada;

O: ortalama,
V: varyans
ssd: standart sapma'dır.

Kabarcık Sıralama, (Bubble Sort)

PDF-XChange Viewer Click to buy NOW! www.docu-track.com

- Kabarcık Sıralama algoritması, dizi sıralama için en yüksek performansı sağlayan bir sıralama yöntemi değildir. Fakat anlaşılması ve uygulaması çok kolay bir sıralama yöntemidir. Nitekim literatürde, bu sıralama yönteminin profesyonel programlarda inanılmaz bir sıklıkla kullanıldığı görülmektedir. Kabarcık Sıralama (Bubble Sort) algoritması, kaynayan suda, büyük kabarcıkların daha hızla yüzeye çıkması model alınarak geliştirilmiştir.

Kabarcık Sıralama

Yöntem son derece basittir. Sıralanacak dizinin üzerinde sürekli ilerlerken her defasında iki öğenin birbiriyle karşılaştırılıp, karşılaştırılan öğelerin yanlış sırada olmaları durumunda yerlerinin değiştirilmesi mantığına dayanır. Döngü başında kontrol değişkenine true değeri atanır. Döngü tüm dizi elemanları sayısının bir eksiğine kadar devam eder. Her eleman bir sonraki ile karşılaştırılır. Her karşılaştırmada eğer elemanın büyüklüğü, bir sonraki elemanın büyüklüğünden daha küçükse, alt sıraya büyük olan üst sıraya taşınır. Taşınma olursa Kontrol değişkeni true değerini alır. Döngü tüm elemanlar sıralanmış oluncaya, yani hiçbir taşınma olmayıncaya kadar devam eder.

Kabarcık Sıralama Algoritmasının Adım Adım İşleyişi

- Dizi "5 1 4 2 8 ", her adımda dizinin **kalın** olarak işaretlenmiş elemanları karşılaştırılan elemanlardır.
- **Birinci Geçiş:**

  (**5 1** 4 2 8) (**1 5** 4 2 8) ilk iki eleman karşılaştırıldı ve yerleri değiştirildi.

  (1 **5 4** 2 8) (1 **4 5** 2 8)

  (1 4 **5 2** 8) (1 4 **2 5** 8)

  (1 4 2 **5 8**) (1 4 2 **5 8**)

İkinci Geçiş:

(**1 4** 2 5 8) (**1 4** 2 5 8)

(1 **4 2** 5 8) (1 **2 4** 5 8)

(1 2 **4 5** 8) (1 2 **4 5** 8)

(1 2 4 **5 8**) (1 2 4 **5 8**)

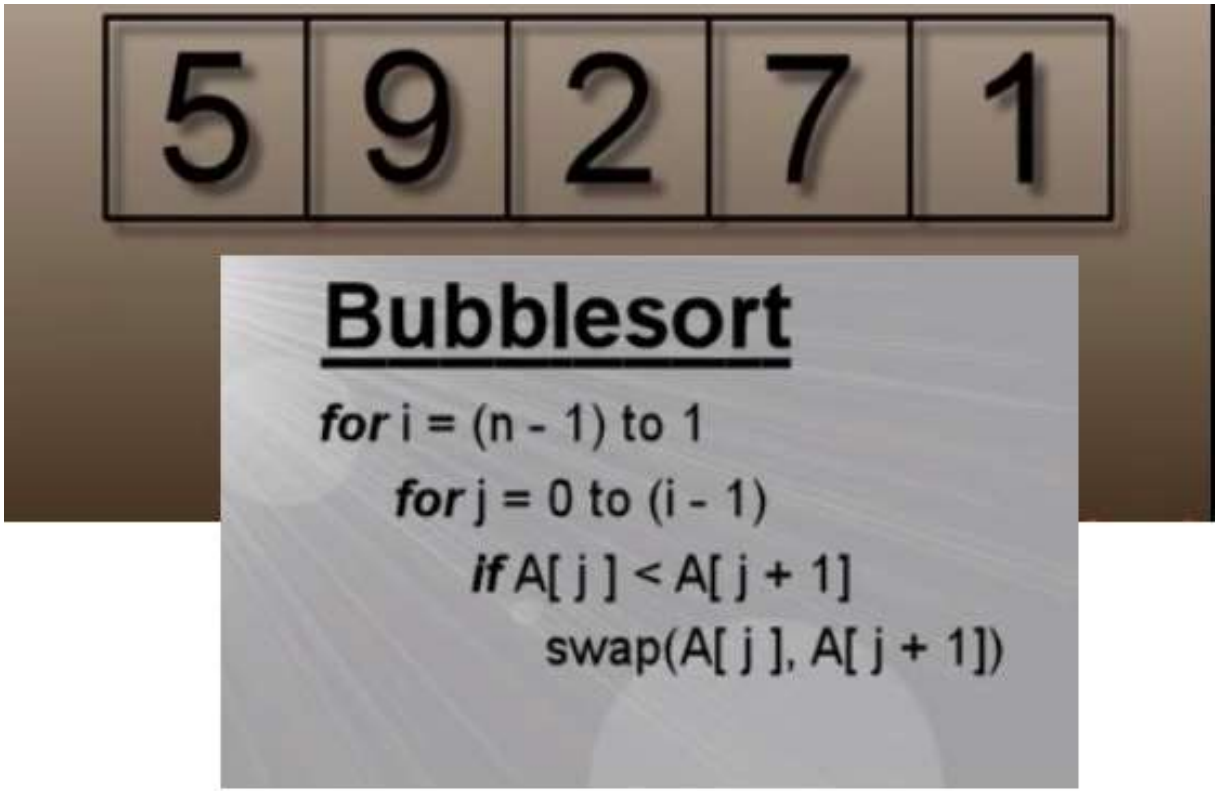
Üçüncü Geçiş:

(**1 2** 4 5 8) (**1 2** 4 5 8)

(1 **2 4** 5 8) (1 **2 4** 5 8)

(1 2 **4 5** 8) (1 2 **4 5** 8)

(1 2 4 **5 8**) (1 2 4 **5 8**)



5	4	2		
4	5	2	1	3
4	2	5	1	3
4	2	1	5	3
4	2	1	3	5
2	4	1	3	5
2	4	1	3	5
2	1	4	3	5
2	1	3	4	5
2	1	3	4	5
1	2	3	4	5
1	2	3	4	5

Küçükten Büyüğe Kabarcık sıralaması


```
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
#define n 10

int kab_sira(int sdizi[]) {
  int i,j,g;
  for(i=n-1;i>0;i--)
        for(j=0;j<i;j++)
           if(sdizi[j]>sdizi[j+1])
               {
                  g=sdizi[j];
                  sdizi[j]=sdizi[j+1];
                  sdizi[j+1]=g;
           }
  return sdizi[n];
}
```

```
void main() {
int dizi[n]={23,56,9,78,96,55,12,
45,456,14};
int i=0;
dizi[n]=kab_sira(dizi);
for(i=0;i<10;i++)
 printf("%d\n",dizi[i]);
 getch();
}
```