6. DİZİLER (arrays) (12nci ve 13ncu haftalar)

Program içinde, benzer tipteki verilerin her birisi için ayrı ayrı değişken tanımlanması ve kullanılması veri sayısı artınca pratik olmamaktadır. Bunun yerine benzer tipteki verileri saklamak ve bunlarla hızlı bir şekilde işlem yapabilmek için **indisli değişkenler**de denilen diziler kullanılır.

6.1 Dizilerin Tanımlanması

Değişkenlerin toplu olarak tanımlanması dizilerle sağlanabilir. Dizilerin boyutları kullanılacak değişken tipine göre değişebilir. Bir ve birden fazla boyutlu diziler, program içinde kullanılmadan önce mutlaka (veri tipi ve boyut olarak) tanımlanmalıdır. Tek boyutlu diziler iki ve daha fazla dizilere göre daha hızlı işlem yapılmasını sağlar.

Örnek 6-1 Aşağıda 7 elemanlı tek duyarlı sayıları saklayabilecek `a` isimli 1 boyutlu dizi tanımlanmıştır : `float a[7];`

Örnek 6-2 12 elemanlı tam sayıları saklayan `b` isimli 1 boyutlu dizi tanımlanmıştır: `int b[12];`

Örnek 6-3 Eleman sayısı daha önceden `maximum` ismi ile tanımlanmış 1 boyutlu dizi tanımlanmıştır: `dizi[maksimum];`

Örnek 6-4 Diziye ilk atamaları tanımlama aşamasında yapılmış ve kesirli sayılar `a` indisli değişkeninde `a[1]=0.1, a[2]=0.2, a[3]=0.3,... a[6]=0.7` şeklinde saklanırlar.

```
float a[7]={0.1,0.2,0.3,0.4,0.5,0.6,0.7};
```

Örnek 6-5 `hafta_gun` isimli diziye haftanın günleri yerleştirilir. `enum` (enumarator) bilgilerin dizi elemanlarına sırayla yerleşmesini sağlar.

```
enum hafta_gun {Pazartesi, Salı, Çarşamba, Perşembe, Cuma, Cumartesi};
```

İhtiyaca göre dizilerin boyut sayısı birden fazla olabilir ve bu tür diziler çok boyutlu diziler olarak adlandırılır.

Örnek 6-6 `a` isimli dizinin 10 satırı ve 10 sütunu vardır. Toplam 100 elemanı vardır. `matris` isimli indisli değişkenin ise 7 satırı ve 4 sütunu vardır. Eleman sayısı 28 dir.

```
const satir = 10,sutun = 10 ;
double a[satir][sutun];
float matris[7][4];
```

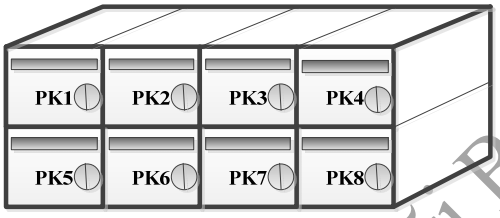
Örnek 6-7 Karakter verilerini dizilerde saklayabilmek için `string.h` ön tanımlayıcısının kullanılması ve indisli değişkenlere yani dizilere ihtiyaç duyulmaktadır. Veri akışı klavyeden bir tuşa dokununca indisli değişkene anında aktarılarak yapılır. Aşağıda `c` isimli ve 5 elemanlı dizinin her elemanına karakterler sırayla aktarılır. Çift tırnaklar arasındaki bilgi 11 elemanlı `tarih` değişkenine teker teker aktarılır.

```
char c[5]={'a', 'b', 'c', 'd', 'e'};
char tarih[11]="01/03/1970";
```

Aynı tipteki veriler bir bölgedeki günlük sıcaklık değerleri olabilir. Bu sıcaklık değerleri 365 gün alınıyorsa 365 elemanlı bir dizi tanımlamak ve sıcaklık değerlerini burada saklamak doğru olacaktır.

Bir sınıftaki öğrencilere ait derslere ait not bilgileri değişik dizilerde saklanabilir. Bir spor dalında takımlara ait takım adları, oynadıkları maçlar, puan değerleri vs. İndisli değişkenlerde saklanabilir.

Diziler, postanelerde veya binalarda kulllanılan posta kutularına benzetilebilir. Herkese ayrılan PK1, PK2, PK3, ... vs isimli kutular aynı büyüklükte ve içine konulacak eşyalar benzer tiptedir.



Şekil 6-1 Dizi tanımı.

Örnek 6-8 Bir kuvvetin harcamış olduğu gücü P=**F**·**v** ile tanımlayabiliriz (denklemdeki **F** kuvvet ve **v** hız vektörleridir, aradaki nokta işareti iki vektörün skaler çarpımını göstermektedir). Bu vektörler **F**=4**i**+3**j**–2**k** Newton ve **v**=4**i**–2**j**+1**k** m/s şeklinde verilmişse gücü, dizi kullanarak hesaplayan bir algoritma ve C/C++ programı aşağıdaki gibi olabilir. Vektörlerin katsayıları/bileşenleri F ve v isimli 3 er elemanlı dizilerde saklanmakta ve skaler çarpıma uygun bir şekilde çarpma işlemi for döngüsü içinde yapılmaktadır. İşlemler tek bir satırdada yapılabilirdi.

Algoritma 6-1 İki vektörün skaler çarpımı.

```
 1.Başla
 2.P=0.0;
 3.F[3]={4.0, 3.0, -2.0};
 4.v[3]={4.0, -2.0, 1.0};
 5.i=0;
 6.Eğer i>2 ise Git 10
 7.P=P+F[i]*v[i];
 8.i=i+1;
 9.Git 6
10.Yaz " Güç = ", P, " Watt";
11.Son.
```

Program 6-1 İki vektörün skaler çarpımı.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <conio.h>
using namespace std;
int main(){
 int i;
 float P=0.0;
 float F[3]={4.0, 3.0, -2.0};
 float v[3]={4.0, -2.0, 1.0};
 for (i=0; i<3; i++) P=P+F[i]*v[i];
```

```
cout << " Güç = " << setprecision(2) << setw(3) <<  P << " Watt" << endl;
getch();
return (0);}
```

Örnek 6-9 Karakter topluluğunun her elemanı dizinin bir elemanına yerleştirilebilir. C/C++ da karakterler dizi elemanlarına birer birer yerleştirilebilir. Bu yüzden her karakter ancak bir kutuya yani indisli bir değişkene konulabilir.

Program 6-2 Karakterlerin dizilere aktarılması.

```
#include <iostream>
#include <conio.h>
using namespace std;
int main(){
/* karakter katarlari için dizi tanimlari */
char adi[10],
     soy[10];
/* karakter katarlarına ilk deger atamaları */
adi[0]='m'; adi[1]='u'; adi[2]='h'; adi[3]='a'; adi[4]='r';
adi[5]='r'; adi[6]='e'; adi[7]='m'; adi[8]='\0';
soy[0]='k'; soy[1]='a'; soy[2]='r'; soy[3]='a';
soy[4]='d'; soy[5]='u'; soy[6]='m'; soy[7]='a';
soy[8]='n'; soy[9]='\0';
cout << "Adı    = " << adi << endl;
cout << "Soyadı = " << soy << endl;
getch();
return 0;}
```

Örnek 6-10 Değişkenlerin veri saklamadan önce içeriklerinin sıfırlanması (tipleri ne olursa olsun) her zaman yararlıdır. Aşağıdaki örnekte `for` döngüsü ile karakterlerin dizi elemanlarına boşluk karakteri aktarılması sağlanmaktadır (`adi[i]='\0';` ve `soy[i]='\0';` işlemleri).

Program 6-3 Karakterlerin dizi elemanlarına aktarılması.

```
#include <stdio.h>
#include <stdlib.h>
main (){ int i;
/* karakter katarlari icin dizi tanimlari */
char adi[10], soy[10];
/* karakter katarlarina ilk deger atamalari */
for (i=1; i<10; i++) adi[i]='\0';
for (i=1; i<10; i++) soy[i]='\0';
printf(" adi    :"); scanf("%s", adi);printf("\n");
printf(" soyadi :"); scanf("%s", soy);
printf(" adi ve soyadi : %s %s\n",adi, soy);
```

```
return EXIT_SUCCESS;}
```

Örnek 6-11 Zamanla değişen bir kuvvet 2 kg lık bir kütleye etkimektedir. I=ΔP momentum değişiminden yararlanarak cismin son hızı $v_{son} = \frac{1}{m}\int_{t_{ilk}}^{t_{son}} Fdt$ integrali ile hesaplanabilir. Bu integral sayısal olarak 1nci derece Simpson yöntemine göre hesaplanabilir. 1nci derece Simpson yönteminde [t_{ilk}, t_{son}] aralığında tanımlı olan bir F(t) fonksiyonunun integrali için, integralin alınacağı aralık önce n (çift sayı) parçaya bölünür (dt=($t_{son} - t_{ilk}$)/n). Burada a=t_{ilk} ve b= t_{son} değerleri integralin başlangıç ve bitiş değerleridir. Bu şekilde herhangi bir F(t) fonksiyonun [a, b] aralığındaki integrali, $I = \frac{1}{m}\int_{t_{ilk}}^{t_{son}} F(t)dt \cong [F(a) + 2F(a+dt) + 2F(a+2dt) + \cdots + F(b)]dt$ şeklindedir. Yukarıdaki bilgilerden yararlanarak bir dosyadan okunan zaman ve kuvvet değerleri kullanılarak bir cismin son hızı aşağıdaki algoritma ve program kullanılarak hesaplanabilir. Çizelgedeki değerler bir dosyada 2 sütun şeklinde saklanmalıdır. Dosyadan okunan bu veriler t ve F ismindeki dizilere aktarılmaktadır. Dosyadan okuma işlemi dosya sonu hatası oluşunca bırakılmakta ve integral işlemine geçilmektedir.

Çizelge 6-1 Zaman bağlı olarak kuvvetin değişimi.

```
t(saniye)|  0  1  2  3  4  5
---------|--------------------
F(Newton)|  0  2  4  4  2  2
```

Algoritma 6-2 Zamana bağlı kuvvetin değişimi (Impuls).

```
 1. Başla
 2. t[] ve F[] dizilerini tanımla;
 3. i=0;
 4. Veri dosyasından değerleri oku, t[i] ve F[i] ye aktar;
 5. Eğer dosya sonu ise Git 8;
 6. i=i+1;
 7. Git 4;
 8. n=i;
 9. i=0;
10. dt=|t[i+1]-t[i]|;
11. I=F[i]+ F[n];
12. i=i+1;
13. Eğer i>=n ise Git 16;
14. I=I+2.0*F[i];
15. Git 12;
16. I=I*dt/(2*m);
17. I değişkenindeki değeri (son hızı) yaz;
18. Son.
```

Program 6-4 Impuls denklemi kullanılarak son hızın hesaplanması.

```
#include <iostream>
```

```
#include <fstream>
int main(){ using namespace std;
    int i=0, n=0;
    float t[6], F[6], a, b, dt, I, m=2;
    ifstream dosya1;  // dosya1 okuma amaçlı tanımlanır
    dosya1.open ("data.txt", ios::in);
    cout << "Veri   t    F \n";
    cout << " No   (s)  (N)\n";
    cout << "--------------\n";
while(dosya1 >> t[i] >> F[i]){ // dosyadan veriler okunur
cout << i << " ) " << t[i] << " " << F[i] << endl;
i++;}  //
n=i-1; // son veri
i=0; // sayaç sıfırlanır
I=F[i]+F[n]; // integral değişkeni
dt=(t[i+1]-t[i]); // zaman aralığı belirlenir
i++;
cout << endl;
while(i < n ){I=I+2.0*F[i]; i++;}
I=(I*dt)/(2*m); // 1nci derece Simpson yöntemine göre integral
  cout << " v = " << I << " (m/s) son hız değeridir" << endl;
    dosya1.close(); // veri dosyası kapatılır
    system("PAUSE");
return 0;
}
```

Örnek 6-12 xy kartezyen koordinat sisteminde konumları (x_i, y_i) ve kütleleri (m_i) ile verilen noktaların oluşturduğu kütleler sisteminin kütle merkezinin konumu (x_c, y_c) şeklinde kartezyen koordinat sisteminde aşağıdaki denklemdeki gibi verilebilir:

$x_c = \frac{\sum_{i=1}^{N} x_i}{M}$ ve $y_c = \frac{\sum_{i=1}^{N} y_i}{M}$

Bu denklemleri kullanarak bir dosyadan okunan verileri (kütle ve konum) indisli değişkenlerde saklayan ve daha sonra bunları kullanarak sistemin kütle merkezini hesaplayan algoritma ve program aşağıda verilmektedir. Burada kullanılan birim sistemi metre ve kilogram cinsindendir (metrik sistem). Programı çalıştırmadan önce aşağıdaki çizelgedeki verileri "`veriler.txt`" isimli bir dosyada saklayınız. Dosyayı, programı derlediğiniz ve çalışır duruma getirdiğiniz klasöre kopyalayınız. Veriler dosyadan bu şekilde okunabilmektedir.

Çizelge 6-2 Değişik konum ve kütle değerlerine sahip cisimlerin küte merkezinin hesabı.

```
Kütle    x     y
(kg)    (m)   (m)
```

```
----------------
0.10   0.5  1.0
1.20   1.0  1.0
0.50   0.9  0.4
0.85   2.0  1.0
```

Algoritma 6-3 Kütle merkezinin belirlenmesi.

1. Başla
2. x[], y[], m[] dizilerini tanımla,
3. i=0,
4. X_C=0,
5. Y_C=0,
6. M=0,
7. x_i, y_i ve m_i değerlerini giriniz,
8. Eğer m_i=0 ise Git 14,
9. M=M+m_i,
10. X_C=X_C+$m_i x_i$,
11. Y_C=Y_C+$m_i y_i$,
12. i=i+1,
13. Git 7,
14. X_C=X_C/M,
15. Y_C=Y_C/M,
16. X_C ve Y_C değişkenlerindeki değerleri yazınız,
17. Son

Program 6-5 Kütle merkezinin hesaplanması.

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string.h>
int main(){ using namespace std;
    int i=0, n=0;
    char s0[10], s1[10], s2[10];
    float x[10],
          y[10],
          m[10],
          M=0,
          Xc=0,
          Yc=0;
    ifstream dosya1;  // dosya1 okuma amaçlı tanımlanır
    dosya1.open ("veriler.txt", ios::out);
    dosya1 >> s0 >> s1 >> s2;
    cout << "    " << s0 << "   "  << s1 << "    "  << s2 << endl;
    dosya1 >> s0 >> s1 >> s2;
```

```
    cout << "    " << s0 << "   " << s1 << "   " << s2 << endl;
    dosya1 >> s0;
    cout << "    " << s0 << endl;
while(dosya1 >> x[i] >> y[i] >> m[i]){ // dosyadan veriler okunur
   cout << i+1 << " ) ";
   cout << setw(4) << fixed <<  setfill(' ') << setprecision(1);
   cout << setiosflags(ios::right);
   cout << x[i] << "  " << y[i] << "  " << m[i] << endl;
   Xc=Xc+x[i]*m[i];
   Yc=Yc+y[i]*m[i];
   M=M+m[i];
   i++;}  //dosyadan okuma işlemi tamamlanır
Xc=Xc/M;
Yc=Yc/M;
cout << endl;
cout << " Kütle merkezi = " << Xc << " , " << Yc << endl;
    dosya1.close(); // veri dosyası kapatılır
    system("PAUSE");
return 0;}
```

Örnek 6-13 Basit harmonik bir titreşkenin kinetik ve potansiyel enerjileri $K=(1/2)mw^2A^2\sin(wt+d)$ ve $U=(1/2)kA^2\cos(wt+d)$ denklemleri ile verilmektedir. Kinetik ve potansiyel enerjilerin zamana göre değişim grafiklerini çizen algoritma ve program aşağıda verilmektedir. Grafik çizimi için kullandığınız derleyicinin bu özelliğinin olup olmadığına bakınız. Yoksa `graphics.h` kütüphanesini sisteminize kopyalayınız. Buradaki grafik çiziminde Bloodshed Software in Developer C++ derleyicisi ve Borland ın grafik kütüphanesi kullanılmıştır (www.bloodshed.net). Ayrıntılar için program satırlarını okuyunuz. Aşağıdaki algoritma programın çalışması konusunda bilgi vermektedir.

Algoritma 6-4 Harmonik titreşkenin kinetik ve potansiyel enerjisinin zamanla değişimi.

```
 1. Başla
 2. m=0.2 kg,
 3. f=1 1/s,
 4. w=2*3.14*f rad/s,
 5. A=2 m,
 6. k=1,
 7. i=0,
 8. t=0 s,
 9. dt=0.1 s,
10. Eğer (t>3) ise Git 16
11. Ki=0.5*m*w*w*A*A*sin(w*t+d)
12. Ui=0.5*k*A*A*cos(w*t-d)*cos(w*t-d)
```

```
13. t=t+dt,
14. i=i+1,
15. Git 10
16. Verileri kullanarak grafik çiziniz,
17. Son.
```

Program 6-6 Harmonik titreşkenin kinetik ve potansiyel enerjisinin zamanla değişimi.

```
#include <cstdlib>
#include <iostream>
#include <graphics.h>
#include <math.h>
#define pi 3.14159
using namespace std;
/* desteklenen fontların adları */
char *font[] = { "DEFAULT_FONT",
                 "TRIPLEX_FONT",
                 "SMALL_FONT",
                 "SANS_SERIF_FONT",
                 "GOTHIC_FONT" };
/* desteklenen text yönelimi isimleri */
char *dir[] = { "HORIZ_DIR",
                "VERT_DIR" };
/* desteklenen yatay text ayarlaması */
char *hjust[] = { "LEFT_TEXT",
                  "CENTER_TEXT",
                  "RIGHT_TEXT" };
/* desteklenen düşey text ayarlanması */
char *vjust[] = { "BOTTOM_TEXT",
                  "CENTER_TEXT",
                  "TOP_TEXT" };
int grafik(int y[100], int x[100], int N){
    int j=0;
    initwindow(700,700); //700x700 grafik penceresi açılır
    setcolor(BLUE);             // Çizim rengi mavi.
    setcolor(COLOR(255,100,0)); // Çizim rengi kırmızı-yeşil.
    setpalette(4, BLUE);        // palet rengi mavi.
    setpalette(4, COLOR(9,9,9));// palet rengi siyah.
    settextjustify(CENTER_TEXT, HORIZ_DIR);
    outtextxy( 40, 250, "Enerji (Joule)");
    outtextxy(550, 520, "Zaman (saniye)");
    moveto(50,50);  // açılan pencerenin sol üst köşesi 0, 0 dır
```

```
    lineto(50,500); // 50, 500 e çizgi çizilir
    lineto(500,500); // 500, 500 e çizgi çizdirilir
    lineto(500,50); // 500, 50 e çizgi çizilir
    lineto(50,50); // 50, 50 e çizgi çizdirilir
    while(j<N){
circle (x[j], y[j], 1);
    j=j+1;
    }
      while(!kbhit());     //kullanıcının bir tuşa basması beklenir
      closegraph();        //grafik penceresi kapatılır
return 0;
     }
int main(int argc, char *argv[])
{
float m=0.2, // kütle (kilogram)
      f=1, // frekans (1/saniye)
      w=2*pi*f, // açısal frekans (radyan/saniye)
      A=2, // genlik (metre)
      t=0, // zaman (saniye)
      k=200, // veri sayısı
      dt=0.05, // zaman aralığı (saniye)
      K, // kinetik enerji değişkeni (Joule)
      d=0; // faz farkı (radyan)
int i=0, N, y[100], x[100];
      while(t<3){
         K=0.5*m*w*w*A*A*sin(w*t+d);
         cout << t << "  " << 500+int(K) << endl;
         y[i]=500+5*int(K);
         x[i]=50+int(t*140);
         t=t+dt;
         i=i+1;
                }
N=i;
grafik(y, x, N);
    system("PAUSE");
    return EXIT_SUCCESS;}
```

Örnek 6-14 Meteoroloji uzmanları bir bölgedeki enlem (batı) ve boylamları (kuzey) verilen 36 noktadaki sıcaklıkları Fahrenheit cinsinden Çizelge 6-3 deki gibi kaydetmişlerdir. Satırlar enlemleri,

sütunlar ise boylamları tanımlamaktadır (soldan sağa veriler batı enlemi, yukarıdan aşağı veriler ise kuzey boylamlarıdır).

Çizelge 6-3 Batı enlemi ile kuzey boylamlarındaki meteorolojik veriler.

	90.0	90.5	91.0	91.5	92.0	92.5
30.0	68.2	72.1	72.5	74.1	74.4	74.2
30.5	69.4	71.1	91.9	73.1	73.6	73.7
31.0	68.9	70.5	70.9	71.5	72.8	73.0
31.5	68.6	69.9	70.4	70.8	71.5	72.2
32.0	68.1	69.3	69.8	70.2	70.9	71.2
32.5	68.3	68.8	69.6	70.0	70.5	70.9

Bu verileri bir dosyadan okuyup ekrana yazan algoritma ve program aşağıda verilmektedir. Bu programla bölgenin ortalama sıcaklığını enlem ve boylamlar için ayrı ayrı hesaplamaktadır.

Algoritma 6-5 Enlem ve boylamlara ait sıcaklık değerlerinin dosyadan okunması.

```
 1. Başla
 2. i=0, j=0;
 3. sicaklik[7][7];
 4. dosya aç("mete.txt","r"); /* dosya okuma amacli acilir */
 5. Eğer i>6 ise Git 14
 6. j=0
 7. Eğer j>6 ise Git 12
 8. sicaklik[i][j]=0.0;
 9. Dosyadan veri Oku ve sicaklik[i][j]) değişkenine aktar
10. j=j+1
11. Git 7
12. i=i+1
13. Git 5
14. Son
```

Program 6-7 Enlem-boylam sıcaklıklarının dosyadan okunup diziye aktarılması.

```
#include "stdafx.h" // MS Visual Studio 2005 için gerekli
#include <iostream>
#include <stdio.h>
#include <conio.h>
int _tmain(int argc, _TCHAR* argv[])
{ // batı enlemindeki değerler sütunlarda
  // kuzey boylamındaki değerler satırlardadır.
int i=0, j=0;
float sicaklik[7][7];
FILE *fin; // dosya işaretci olarak tanımlanır
fin = fopen("mete.txt","r"); /* dosya okuma amacli acilir */
for (i=0; i<=6; i++)
for (j=0; j<=6; j++) sicaklik[i][j]=0.0;
```

```
printf("     ");
for (j=0; j<=5; j++){ // ilk satırda 6 değer var
     fscanf(fin, "%f", &sicaklik[0][j]);
     printf("%4.1f ", sicaklik[0][j]);}
printf("\n");
i=1;
while (!feof(fin)) // dosya sonuna kadar okuma yapılır
{ for (j=0; j<=6; j++) {
  fscanf(fin, "%f", &sicaklik[i][j]);
  if (!feof(fin)) {
// dosya sonu durumunda ekrana birşey yazdırılmaz
  if (j==0) printf("%2.1f ", sicaklik[i][j]);
  else printf("%4.1f ", sicaklik[i][j]);}
  } // for döngüsü sonu
  printf("\n");
  i=i+1;}
  getch();
  return 0;}
```

6.2 SORULAR

1. Bir botun hızı 10 m/s ye ulaştıktan sonra motoru durdurulup kıyıya yanaştırılmak istenmektedir. Bu süre içinde botun kenara ulaşmasını ifade eden denklem $v(t) = v_0 e^{-ct}$ ile verilmektedir. Burada $v(t)$, t anındaki hızı, v_0 ilk hızı (botun motorunun durdurulduğu andaki hızı) ve c bir sabittir. Motor durdurulduktan sonraki t=20 saniye sonunda botun hızı 5 m/s ye düşmüştür. Bu verileri kullanarak denklemdeki *c* sabitini hesaplayan bir algoritma geliştiriniz ve program yazınız (eşitliğin her tarafının logaritması alınır : $\ln(v(t)) = \ln(v_0) - ct$, 5 m/s ye düşmesi için geçen süre 20 saniye ise denklem $\ln(5) = \ln(10) - c\,20$ olur ve *c* nu son denklemden çözülür).