6. DİZİLER

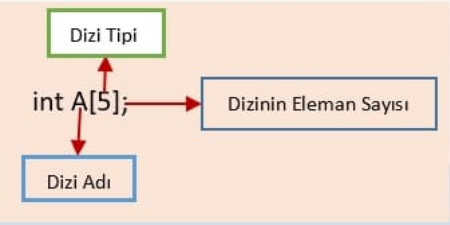
Giriş

Programlama dillerinde, aynı tipte veri parçalarını bir araya getirip gruplayan veri yapılarına dizi denir. Bir dizi içinde gruplanmış veri parçaları, *int* veya *float* gibi temel veri tiplerinden biri olabilir. Bu bölümde *int* ve *char* gibi temel veri tipindeki dizileri ele alacağız.

Diziler birkaç veriyi veya binlerce veriyi gruplayıp saklayabilir. Bu açıdan dizler, yedinci bölümde ele aldığımız yapılara benzer. Ancak, dizilerle yapılar arasında önemli iki fark bulunmaktadır. Bu farklardan birincisi, yapılar farklı tipteki verileri gruplayabilir, diziler ise aynı tipteki verileri gruplamak için kullanılır. Yapılarla diziler arasındaki ikinci fark, yapının üyelerine isimle ulaşılırken, dizi elemanlarına bir indeks ile ulaşılır.

6.1. Tek Boyutlu Dizileri tanımlamak

Değişkenlerde olduğu gibi, dizileri de kullanmadan önce tanımlamak gerekir. Bir dizi tanımlanırken, önce dizinin depolayacağı verilere bağlı olarak dizinin tipi belirlenir. Diziye bir isim verilir ve bilgisayar programlarında bu isimle dizi elemanlarına ulaşılır. Tek boyutlu dizilerde, bir köşeli parantez içerisine dizinin eleman sayısı yazılır. Köşeli parantez içerisine yazılan sayı ya sabit bir sayı olmalıdır ya da değeri sabit olan bir deyim olmalıdır. Ayrıca, bu sayı tam sayı olmak zorundadır. Beş adet tamsayı tutacak A dizinin söz dizimi şekil 6.1'de gösterilmiştir.



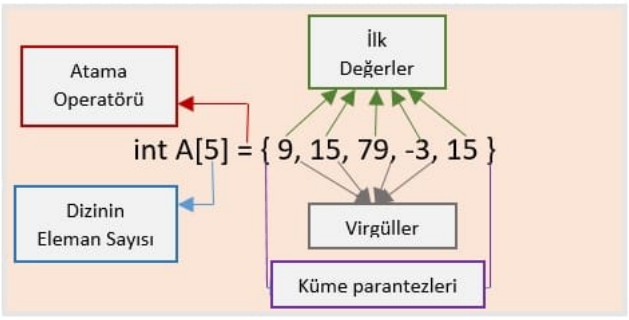
Şekil 6.1 Dizi tanımının söz dizimi

Beş elemanlı bu A dizisinin elemanları sırasıyla A[1], A[2], A[3],A[4], A[5] şeklinde ifade edilir. *Dizi, art arda gelen aynı tip verileri saklayan bellek elemanları veya bellek hücreleri olarak tanımlanır.* Bir dizi bellekte sabit büyüklükte bir yere yerleşir. Dizi tanımlandıktan sonra program içerisinde dizinin bellekte kapladığı alan miktarında artma veya azalma yapmak mümkün değildir.

Örneğin, 100 elemanlı bir dizi tanımlanırsa, program içerisinde bu dizinin sadece 10 elemanı kullanılsa bile geri kalan 90 elemanı bellekte yer işgal etmeye devam eder. Bu nedenle diziler tanımlanırken, dizilerde saklanılacak veri miktarı iyi analiz edilmelidir.

6.2. Tek Boyutlu Dizileri İlk Kullanıma Hazırlamak

Bir dizi tanımlandığı sırada dizi elemanlarının her birine bir değer atanabilir. Bu işleme diziyi ilk kullanıma hazırlama işlemi denir. Şimdi, yukarıda Dizileri Tanımlama bölümünde tanımı yapılan *int A[5];* dizisini ilk kullanıma hazırlayalım. Dizileri ilk kullanıma hazırlama işleminin söz dizimi aşağıda *şekil 6.2'de* gösterilmiştir.



Şekil 6.2 Diziyi ilk Kullanıma Hazırlama

İnt A[5]; dizisi beş elemanı vardır. Şekil 6.3'ten de görüldüğü gibi, dizinin birinci elemanının indisi sıfırdır. Beş elemana sahip bu dizinin son elemanının indisi de dörttür. Bu biraz akıl karışıklığına neden olabilir. Akıl karışıklığından kurtulmak için, tek boyutlu dizilerde dizinin son elamanının indisinin, dizinin eleman sayısından bir eksik olduğunu aklımızda tutmak yararlı olabilir. Bu bölümün ilerleyen kısımlarında, dizinin elemanlarına ulaşmak istediğimizde sürekli indisleri kullanacağız. Şekil 6.3'te dizi elemanlarının bellekte sembolik olarak yerleşimi gösterilmiştir.

Şekil 6.3 int A[5]={9,15,79,-3,15} dizisinin elemanları

6.2.1. Program 6.1

int A[5]={9,15,79,-3,15} dizisi verilmiştir.

a-) C++ kodunu yazınız?

b-) C++ kodunu yazınız?

c-) Diziye elman atama işlemi yapmadan(, dizi elemanlarını ekrana yazdıran algoritmanın C++ kodunu yazınız?

d-) Beş elemanlı int A[5]={9,15,79}; dizisine üç eleman atayıp, dizi elemanlarını ekrana yazdıran algoritmanın C++ kodunu yazınız?

e-) int A[5]={0}; ataması yaparak dizi elemanlarını ekrana yazdıran algoritmanın C++ kodunu yazınız?

f-) int A[5]={9,15,79,-3,15,6}; dizisine belirlenen eleman sayısından fazla eleman atayınız ve C++ uygulamasını yeniden yazınız?

Çözümler:

a-)

Daha önce for döngülerini incelemiştik. Hatırlanacağı gibi biz, for döngülerini daha çok döngüye girmeden, döngünün kaç defa dönmesi gerektiğini bildiğimiz durumda kullanıyorduk. Problem 6.1'de verilen A[5]= {9,15,79,-3,15} dizisinin eleman sayısı beş olduğuna göre dizinin elemanlarının tamamının ekrana yazdırılabilmesi için döngünün beş defa dönmesi gerektiğini öngörmek zor değildir. Aşağıda verilen C++ kodundan da anlaşılacağı gibi, döngünün ilk dönüşünde dizinin sıfır indisine sahip elamanı ekrana yazdırılır. Döngünün İkinci dönüşünde ise, dizinin ikinci elamanı ekrana yazdırılır ve bu işlem dizinin son elemanının ekrana yazdırılmasına kadar devam eder.

```
#include <iostream>
using namespace std;
int main() {

    int A[5]={9,15,79,-3,15};
        for(int i=0;i<5;i++)
        cout<<"   "<<A[i]<<endl;
    return 0;
}
```

```
9
15
79
-3
15
```

Program 6.1-(a)'nın C++ kodu *Ekran çıktısı*

b-)

Dizi, hem tanım olarak, hem de dizi elemanlarına atanan değerler açısında bu problemin (**a)** şıkkının çözümünde kullanılan dizi ile aynıdır. Buradaki farklılık dizinin elemanlarının tersten yazdırılmasının istenmesi. Çözüm için önce, dizinin son elemanının(***A[4]****'ün*) yazdırılması, sonra sondan bir önceki elemanın(***A[3]****'ün*) yazdırılması ve bu işlemin ilk eleman(**A[0]**) yazdırılana kadar bu şekilde devam etmesi gerekiyor.

A[5]={9,15,79,-3,15} dizisinin elemanlarının tersten yazdırılması için Problem 6.1'in (**a**) şıkkı için yazılan C++ kodunda, for parantezinde bazı değişiklikler yapılması gerekmektedir. Yapılması gereken değişikliklerden ilki, döngünün başlangıcının dizinin son elemanına erişecek şekilde düzenlenmesi (**i=4**), ikincisi, Programın hangi koşul altında döngüdeki ifadeleri çalıştıracağını belirlemek(**i>=0**) ve son olarak artış değerini istenen çözüme göre yeniden düzenlemek. Bu problemin çözümü için artış değeri ***i--***olarak belirlenmiştir.

6.3.1. (b)'nın çözümü için yazılması istenilen C++ kodu, yukarıda açıklaması yapılan değişiklikler dikkate alınarak, yeniden yazılmış ve aşağıda ekran çıktısı ile birlikte gösterilmiştir.

```
#include <iostream>
using namespace std;
int main() {

    int A[5]={9,15,79,-3,15};
    for(int i=4;i>=0;i--)
        cout<<"   "<<A[i]<<endl;
    return 0;
}
```

```
15
-3
79
15
9
```

Program 6.1-(b)'nın C++ kodu *Ekran çıktısı*

c-)

Problem 6.1'in (a) şıkkı için yazılan programdan dizi elemanlarına değer atama bölümü kaldırılmıştır. Programda for parantezinden önce sadece dizinin tanımı(***int A[5];***) bırakılmıştır. Program çalıştırıldıktan sonra, ekran çıktısı olarak rastgele değerler çıktığı gözlenmiştir. Bunun nedeni, Dizi tanımlandığında, tanımlanan dizi için bellekte yer ayrılır. Fakat dizi elemanlarına değer atanması yapılmadığı için dizi için, ayrılan adreslerdeki rasgele sayılar çıktı olarak üretilir. 6.3.1. (c)'nın çözümü için yazılması istenilen C++ kodu, yukarıda açıklaması yapılan değişiklikler dikkate alınarak, yeniden yazılmış ve aşağıda ekran çıktısı ile birlikte gösterilmiştir.

```
#include <iostream>
using namespace std;
int main() {

    int A[5];
        for(int i=0;i<5;i++)
        cout<<"   "<<A[i]<<endl;
    return 0;
}
```

Program 6.1-(c)'nın C++ kodu

```
1
0
4254425
0
86
```

Ekran çıktısı

d-)

int A[5]; dizisini ilk kullanıma hazırlarken eksik sayıda eleman atanırsa (**int A[5]={ 9, 15, 79**}), atama yapılmayan elemanlara otomatik olarak sıfır değeri atanır. 6.3.1. (d)'nın çözümü için yazılması istenilen C++ kodu, yukarıda açıklaması yapılan değişiklikler dikkate alınarak, yeniden yazılmış ve aşağıda ekran çıktısı ile birlikte gösterilmiştir.

```
#include <iostream>
using namespace std;
int main() {

    int A[5]={9,15,79};
        for(int i=0;i<5;i++)
        cout<<"   "<<A[i]<<endl;
    return 0;
}
```

Program 6.1-(d)'nın C++ kodu

```
9
15
79
0
0
```

Ekran çıktısı

Program 6.1-(d)'nın C++ kodu Ekran çıktısı

e-)

Eleman sayısından bağımsız olarak, bir diziye sadece bir adet sıfır(***{0}***) ataması yapılırsa o dizinin bütün elemanlarının değeri sıfır olur. Problem 6.1'de ele aldığımız örnek diziye de ***int A[5]={0};*** şeklinde bir atama yapılmış ve programın ekran çıktısında dizinin bütün elemanlarının sıfır olduğu gözlenmiştir.

```
#include <iostream>
using namespace std;
int main() {

    int A[5]={0};
        for(int i=0;i<0;i++)
        cout<<"   "<<A[i]<<endl;
    return 0;
}
```

Program 6.1-(e)'nın C++ kodu

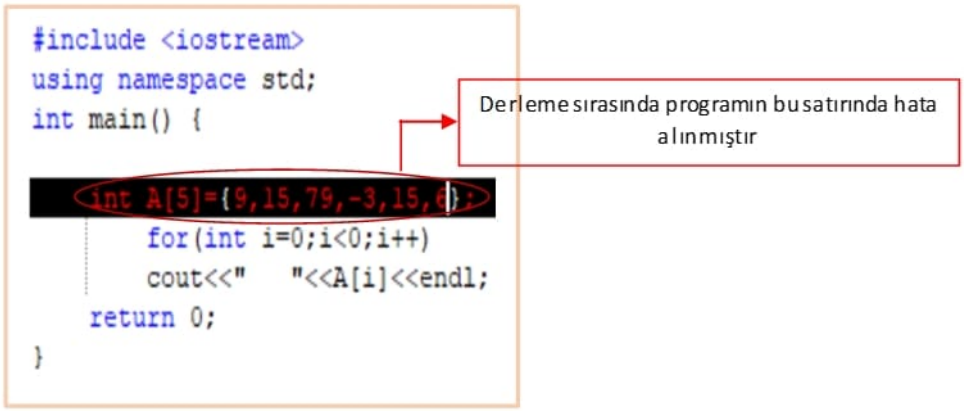
```
0
0
0
0
0
```

Ekran çıktısı

Program 6.1-(e)'nın C++ kodu Ekran çıktısı

f-)

Bir dizinin ilk kullanıma hazırlanması sırasında diziye, tanımında belirlenen eleman sayısında daha fazla değer ataması yapılırsa program derleme sırasında hata verir. Problem 6.1'de ele aldığımız örnek diziye de ***int A[5]={ 9,15,79,-3,15,6};*** şeklinde bir atama yapılmış ve programın derlenmesi sırasında hata ile karşılaşılmış ve herhangi bir ekran çıktısına ulaşılamamıştır.



Program 6.1-(f)'nın C++ kodu

6.3. Tek Boyutlu Dizi Elemanlarına Klavyeden Değer Girmek

6.3.'te verilen dizi örneğinde, dizi elemanlarının değerleri program içerisinde atanmıştı. Ancak, birçok problemin çözümünde, dizi elemanlarının değerlerinin klavyeden girilmesi gerekebilir. C++'ta dizi elemanlarına klavyeden değer girilmesi, değişkenlerin değerlerinin klavyeden girilmesine çok benzer.

cin>>A[0];

Dizilerin elemanlarına, klavyeden değer girilmesi için önce dizinin adı ve sonra da köşeli parantezin içinde, dizinin hangi elemanına değer girilecekse, o elemanın indisinin bilinmesi gerekir.

Bir dizinin bütün elemanlarına değer girilebilmek için ya dizinin her elanı için ayrı ifade yazmamız gerekecek(1. Yöntem) ya da bunu daha kısa yoldan bir for döngüsü içerisinde yapacağız(2. Yöntem). Bu iki örnekte aşağıda gösterilmiştir.

```
#include <iostream>
#include <locale.h>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish");
    int A[5];
    // Dizinin elemanlarına klavyeden değer girmek için gerekli ifadeler
        cout<<"Dizinin Birinci  elemanı: "; cin>>A[0];
        cout<<"Dizinin İkinci   elemanı: "; cin>>A[1];
        cout<<"Dizinin Üçüncü   elemanı: "; cin>>A[2];
        cout<<"Dizinin Dördüncü elemanı: "; cin>>A[3];
        cout<<"Dizinin Beşinci  elemanı: "; cin>>A[4];

    // Dizinin elemanlarına girilen değerleri göstermek için gerekli ifadeler
        cout<<endl<<endl;
        cout<<A[0]<<endl;
        cout<<A[1]<<endl;
        cout<<A[2]<<endl;
        cout<<A[3]<<endl;
        cout<<A[4]<<endl;

    return 0;
}
```

1. *Yöntem. Dizi elemanlarına klavyeden değer girilmesi*

İki programda daire içerisine alınan ifadeler aynı işlemi yapmaktadır

```
#include <iostream>
#include <locale.h>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish");
    int A[5];
    // For döngüsü İle dizi elemanlarına değer girilmesi
    for(int i=0;i<5;i++)
    {
        cout<<i+1<<". Eleman : ";cin>>A[i];
    }
    /* For döngüsü ile dizinin elemanlarına klavyeden girilen
       degerlerin gosterilmesi*/
    for(int i=0;i<5;i++)
    {
        cout<<"  "<<A[i]<<endl;
    }
    return 0;
}
```

2. *Yöntem Dizi elemanlarına klavyeden for döngüsü kullanılarak değer girilmesi*

Yukarıda dizi elemanlarına değer girilmesi için yazılan iki C++ programı da aynı işlemi yapmaktadır. İncelendiğinde kolayca görülebileceği gibi, 1. Yöntemde dizinin bütün elemanlarına giriş yapabilmek için çok daha fazla ifade yazılmıştır. Dizinin eleman sayısı arttığında bu ifadeler de eleman sayısı kadar artacaktır. Örneğin 1000 elemanı olan diziye klavyeden değer girilebilmesi için bu yöntem kullanılırsa 1000 ifadeye ihtiyaç duyulacaktır.

Girişler için for döngüsü kullanılırsa(2. Yöntem) Tek bir ifade ile dizinin bütün elemanlarına klavyeden değer girişi yapılabilir. Bu yöntemde dizinin eleman sayısının artması programa yeni ifadeler eklenmesini gerektirmez. Bunun için for parantezindeki koşul ifadesinin değiştirilmesi yeterli olacaktır. Örneğin 1000 elemanlı bir dizinin bütün elemanlarına değer girebilmek için koşul ifadesinin ***i<1000*** şeklinde değişmesi yeterli olacaktır.

6.4. Tek Boyutlu Dizi Elemanlarının Ortalamasını Almak

Bu uygulama, bir okulda derslere ait sınıf ortalamalarının hesaplanması için, C++ dili ile yazılmıştır. Uygulama çalıştırıldığında kullanıcıdan önce ortalaması hesaplanacak dersin adının, sonra sınıftaki öğrenci sayısının, daha sonra da öğrencilerin dersten aldıkları notların girilmesi istenmektedir. Bu bölümde daha önce ele alınan uygulamalarda dizilerin eleman sayıları sabit olarak girilmekteydi. Bu uygulamada ise kullanıcıdan dizinin eleman sayısının(ogrSayi) klavyeden girilmesi istenmektedir. Uygulamada yapılan bu değişiklik, programa önemli bir esneklik kazandırmış ve öğrenci sayısının farklı olduğu derslerin ortalamasının koda müdahale etmeden alınması sağlanmıştır. Buna ilave olarak, ders ismi girişinin kullanıcıdan alınması da uygulamaya ek esneklik kazandırmıştır.

```
#include <locale.h>
using namespace std;
int main()
{
    setlocale(LC_ALL,"Turkish");
    int ogrSayi;
    string ders;
    cout<<"Ortalaması Alınacak Dersi Giriniz : ";
    cin>>ders;
    cout<<"Öğrenci Sayısı Giriniz            : ";
    cin>>ogrSayi;
    int dersNotu[ogrSayi];
    for (int i = 0; i < ogrSayi; i++)
    {
        cout <<i + 1<<". ogrencinin "<<ders<<" Notunu girin: ";
        cin >> dersNotu[i];
    }
    int toplam = 0;
    for (int i = 0; i < ogrSayi; i++) toplam += dersNotu[i];
    float sinif_ort = toplam / ogrSayi;
    cout << "Sinif ortalamasi: " << sinif_ort << endl;
    system("PAUSE");
    return 0;
}
```

Dizinin eleman sayısı klavyeden girilerek uygulamaya esneklik kazandırılmıştır.

Öğrencilerin derslerden aldıkları notların ortalamasını hesaplayan program

```
Ortalaması Alınacak Dersi Giriniz : Matematik
Öğrenci Sayısı Giriniz            : 2
1. ogrencinin Matematik Notunu girin: 37
2. ogrencinin Matematik Notunu girin: 54
Sinif ortalamasi: 45.5
```

```
Ortalaması Alınacak Dersi Giriniz : Fizik
Öğrenci Sayısı Giriniz            : 4
1. ogrencinin Fizik Notunu girin: 67
2. ogrencinin Fizik Notunu girin: 29
3. ogrencinin Fizik Notunu girin: 53
4. ogrencinin Fizik Notunu girin: 74
Sinif ortalamasi: 55.75
```

Programın iki farklı derse ait ortalama hesabı ekran çıktısı

6.5. Çok Boyutlu Diziler

Bazı problemin çözümünde tek boyutlu diziler yetersiz kalabilir. Bu nedenle pek çok programlama dili birden fazla boyuttan oluşan dizilerin kullanılmasına izin verir ve bu şekilde tek boyutlu dizilerle çözümü modellenemeyen birçok problem iki veya daha fazla boyuta sahip dizilerle çözülebilir. Ayrıca, matematik derslerinden aşina olduğumuz matris işlemlerinin çözümü için de yine programlama dillerinin sağladığı bu özelliklerden yararlanılır.

İki boyutlu bir matris kullanılarak çözülebilecek problemlere örnek teşkil etmesi için, bir pazarlama firmasının üç farklı bölgede ürünlerini sattığını ve bu firma aylık olarak bu bölgelerde satış miktarlarını, bilgisayarda, bir program aracılığı ile takip etmek istediğini varsayalım. Bu problemin çözümü için, hem bölge hem de ay bilgisi tutulması gerektiğinden, yani tutulması gereken bilginin tablo karakteri göstermesinden dolayı, iki boyutlu bir matris kullanılması gerektiğinin tahmin edilmesi zor değildir. İki veya daha çok boyutlu matrisler, bu ve benzeri problemlerin çözümü için kullanılabileceği gibi iki matrisin toplanması, matrisin transpozunun alınması, birim matris oluşturulması ve benzer matris işlemlerinin yapılmasında da yaygın olarak kullanılır.

6.6. Çok Boyutlu Dizilerin Tanımlanması

Çok boyutlu dizilerin tanımı temelde tek boyutlu dizilerin tanımına benzer. İki tanımlama arasındaki temel farklılık, tek boyutlu dizilerde indis sayısı bir olmasına karşılık, örneğin iki boyutlu dizilerde indis sayısı ikidir. İki boyutlu dizinin birinci indisi satırlara ikinci indisi de sütunlara karşılık gelir. Bilgisayar hafızasında(RAM) tek boyutlu bir yapı vardır ve bundan dolayı çok boyutlu yapılar hafızada tek boyuta indirgenerek saklanır. Dolayısı ile iki boyutlu dizilerde birinci indisin satır, ikinci indisin sütün olduğunu söylemek bir kabulden ibarettir. Kolayca tahmin edilebileceği gibi, üç boyutlu dizilerde de indis sayısı üçtür.

Aşağıda C++ yazım kurallarına göre tanımlanmış bir matris gösterilmiştir:

```
int A[2][3];
```

Bu ifadeyi C++ programında yazarsak hafızada 2x3 boyutunda bir matris yer açılır. Açılan bu yer altı hücrelidir ve bu hücrelerde altı farklı tamsayı tutulabilir.

Aşağıda 2x3 matrisin satır ve sütun indisleri gösterilmiştir.

A[0][0] A[0][1] A[0][2]
A[1][0] A[1][1] A[1][2]

Aşağıda dizinin bütün elemanlarına değer atanmış ve atanan değerler tabloda gösterilmiştir.

```
A[0][0]=12;
A[0][1]=15;
A[0][2]=27;
A[1][0]=3;
A[1][1]=83;
A[1][2]=9;
```

12	15	27
3	83	9

6.6.1. Problem 6.2

ogrNot[4][2] şeklinde İki boyutlu bir dizi tanımlayın ve bu dizinin her bir satırındaki dizi elemanlarına bir öğrencinin vize ve final notunu atayınız. Daha sonra bu dört öğrencinin ağırlıklı not ortalamasını hesaplatıp ekrana yazdıran bir C++ programı yazınız?

```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    // Çok boyutlu diziye ilk değer veriliyor
    int ogrNot[4][2] = { {15, 30}, {45, 77}, {38, 55}, {100, 90} };

    for (int i = 0; i < 4 ; i++)
    {
        float ort = ogrNot[i][0] * 0.4F + ogrNot[i][1] * 0.6F;

        cout << i + 1 << ". ogrencinin ortalamasi: "<<ort<<endl;
    }
    system("PAUSE");
    return 0;
}
```

İki Boyutlu dizi ilk kullanıma hazırlanıyor.

Ortalama hesaplanıp, ekrana yazdırılıyor

Dört öğrencinin not ortalamalarını hesaplayan program

```
1. ogrencinin ortalamasi: 24
2. ogrencinin ortalamasi: 64.2
3. ogrencinin ortalamasi: 48.2
4. ogrencinin ortalamasi: 94
Press any key to continue . . .
```

Öğrencilerin not ortalamalarını hesaplayan programın çıktısı

6.6.2. Problem 6.3

Birim matris oluşturan algoritmanın C++ kodunu yazınız ve akış şemasını çiziniz?

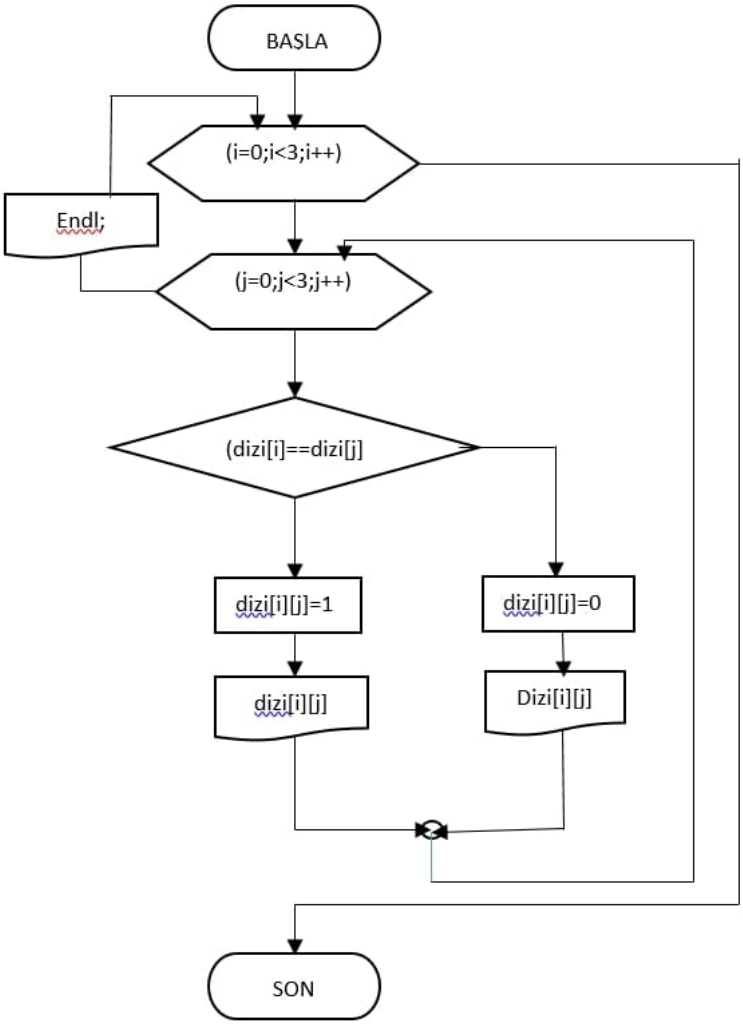
```
#include <iostream>
using namespace std;

int main() {
    int dizi[3][3],i,j;
    for(i=0;i<3;i++){
        for(j=0;j<3;j++){
            if(i==j){
             dizi[i][j]=1;
             cout<<"  "<<dizi[i][j];
            }
            else
            {
             dizi[i][j]=0;
             cout<<"  "<<dizi[i][j];
            }
        }
        cout<<endl;
    }
        return 0;
}
```

```
1  0  0
0  1  0
0  0  1
```

Problem 6.3'ün ekran çıktısı

Şekil 6.4 oldukça ilginç bir örnektir. Akış şemasından da görülebileceği gibi, şekilde iç içe iki döngü vardır. Birinci döngü(dıştaki döngü) satırları oluştururken, ikinci döngü(içteki döngü) sütunları oluşturmaktadır. Birinci döngüde i'nin değeri sıfır olunca akış ikinci döngüye geçmektedir. İkinci döngüye ilk girişte, J'nin değeri de sıfır olduğu için dizi[i][j]=1 ifadesi çalıştırılmış ve arkasından bu değer ekrana yazdırılmıştır. Sonra içteki döngü bir kere daha çalışmış, fakat bu defa i'nin değeri hala sıfır iken j'nin değeri bir olmuştur i ve j'nin eşitliği sağlanmadığı için karar bloğunun else kısmındaki ifade çalışmış ve ekrana sıfır yazılmıştır. i'nin değeri sıfırken, j iki olduğunda, içteki döngü üçüncü ve son defa çalıştırılmış ve ekrana tekrar sıfır yazdırılmıştır. J, üç olduğunda matrisin ilk satırı tamamlanmıştır ve akış yeni satıra geçilerek, dıştaki döngüye geçmiştir. Akış dıştaki döngüye tekrar geldiğin de i'nin değeri bir olmuş ve akış tekrar içteki döngüye geçmiş ve içteki döngüye ilk girişte, j'nin değeri sıfır olduğu için karar bloğunun else kısmındaki ifadeler çalıştırılmış ve ikinci satırın il elemanı olarak ekrana sıfır yazdırılmıştır. Döngülerin çalışması tamamlanınca ekrana 3x3 bir birim matris yazdırılmıştır.



Şekil 6.4 Problem 6.3'ün Akış şeması

Bölüm Özeti

Bu bölümde yapısal programlarda tek boyutlu ve çok boyutlu dizilerin kullanım amaçlarını ve değişkenlere göre avantajlı yanlarını öğrendik.

Ayrıca, hem tek boyutlu dizileri hem de çok boyutlu dizileri ilk kullanıma hazırlamayı, dizi elemanlarına klavyeden değer girmeyi, dizi elemanlarının değerlerini bellekten okuyup hesaplama yapmayı öğrendik

İki boyutlu dizi örneğinden yola çıkarak içerisinde dizi olan algoritmaların akış şemalarını çizmeyi ele aldık.

H. Burak Tungut, Algoritma ve Programlama Mantığı, KODLAB Yayın Dağıtım Yazılım ve Eğitim Hizmetleri San. ve Tic. Ltd. Şti, 2019.

Duygu Arbatlı Yağcı, Nesne Yönelimli C++ Programlama Kılavuzu, Alfa Basım Yayım Dağıtım Ltd. Şti, 2016.

G. Murat Taşbaşı, C programlama, Altaş yayıncılık ve Elektronik Tic. Ltd. Şti. 2005.

Muhammed Mastar, Süha Eriş, C++, KODLAB Yayın Dağıtım Yazılım ve Eğitim Hizmetleri San. ve Tic. Ltd. Şti, 2012.

Sabahat Karaman, Pratik C++ Programlama, Pusula Yayınevi,2004