9. YAPILAR (STRUCTURES)

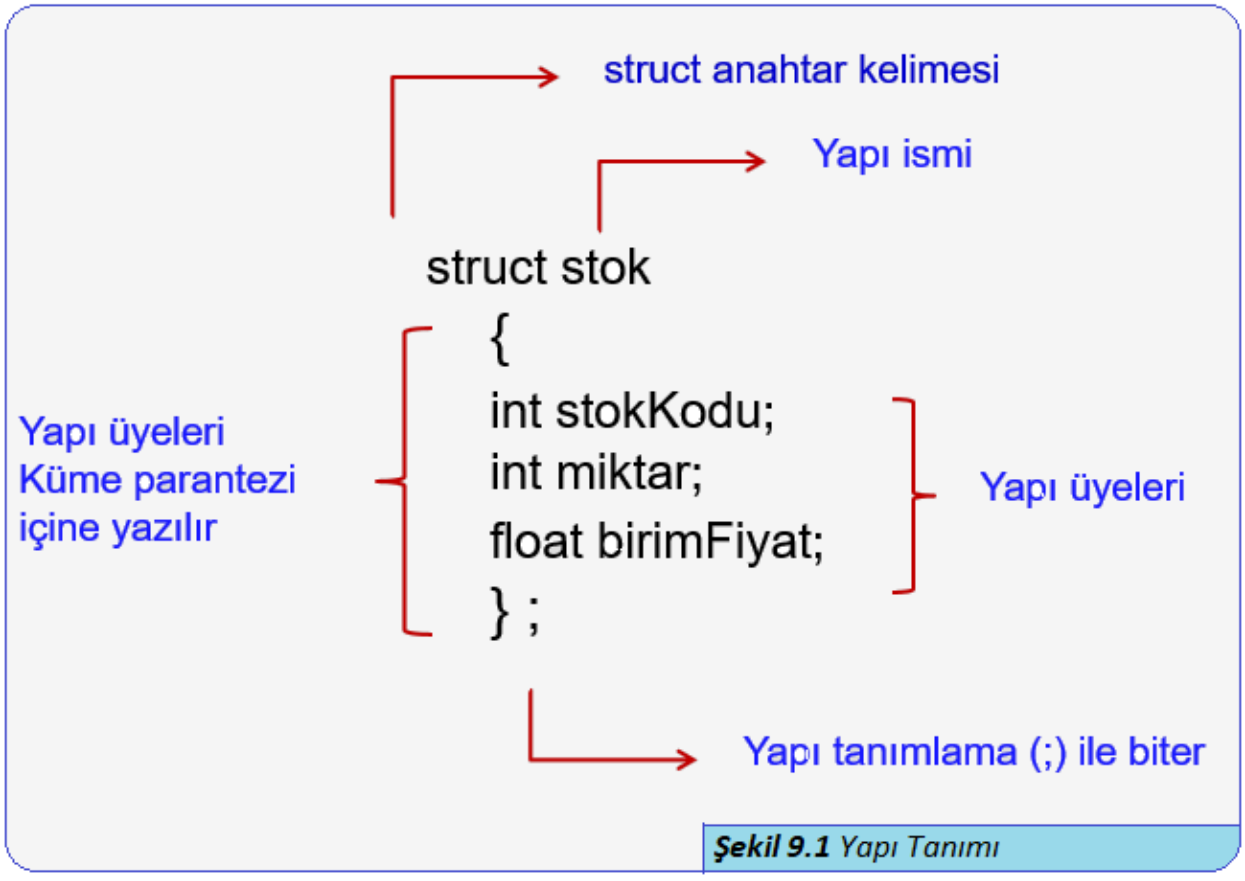
Giriş

Bundan önceki bölümlerde int, float, double, char vb. C++' ın desteklediği temel veri tiplerini kullanmıştık. Hatırlanacağı gibi, temel veri tipleri kullanılarak tanımlanan değişkenler tek bir bilgiyi temsil edebiliyordu. Bu bölümde temel değişkenleri bir araya getirerek kendi veri tiplerimizi oluşturacağız ve oluşturulan bu veri tipleri ile birden fazla verinin nasıl temsil edileceğini öğreneceğiz. C++'ta yapı(struct) kendi veri tiplerimizi oluşturmanın araçlarından biridir.

Yapı(struct), birbiri ile ilişkili verileri bir isim altında toplayarak tanımlamak olarak tarif edilebilir. Bu şekli ile yapılar dizilere benzer. Diziler gibi birden fazla veriyi tutabilir ve yine diziler gibi bellekte sıralı şekilde bulunurlar. Ancak, diziler Altıncı bölümden de vurguladığımız gibi, aynı tür verileri bir isim altında temsil etmek için kullanılıyordu. Yapılar ise, aynı türdeki verileri bir isim altında toplayabildiği gibi, farklı türdeki verileri de aynı isim altında toplayabilirler. Bunu başka bir şekilde ifade etmek gerekirse; bir yapının altında tanımlanan değişkenler aynı tipte olmak zorunda değildir. Örneğin bir değişken int, diğeri float ve bir diğeri de char tipinde olabilir.

9.1. Yapıların Oluşturulması ve Kullanılması

Bir yapının tanımı **struct** anahtar kelimesi ile başlatılır. Daha sonra yapıya bir isim verilir ve küme parantezi içerisinde yapının üyeleri tanımlanır. Yapıların tanımında kullanılan kapanış küme parantezinin sonuna, kararlardan, döngülerden ve fonksiyonlardan farklı olarak noktalı virgül konulur. Şekil 9.1'de yapı tanımının söz dizimi gösterilmiştir.

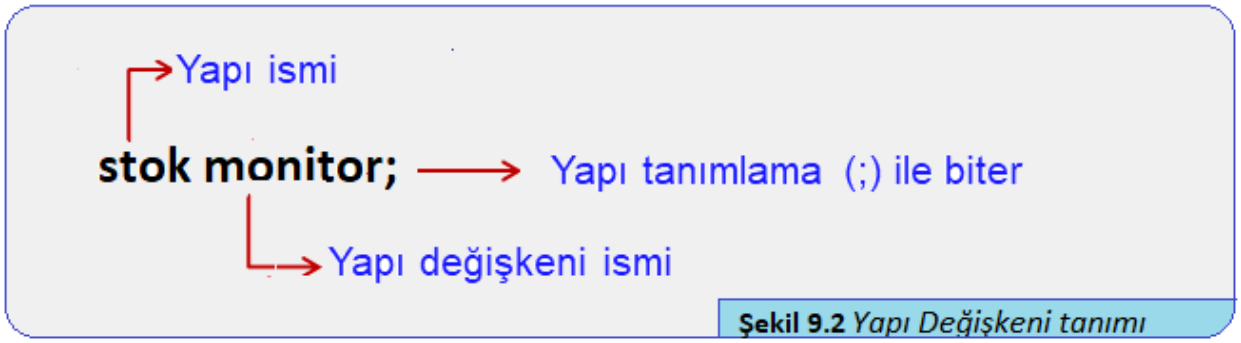


Şekil 9.1 Yapı Tanımı

Şekil 9.1'de tanımı yapılan yapının, ikisi (*stokKodu*, *miktar*) int ve biri de float (*birimFiyat*) olmak üzere üç üyesi vardır. Bu tanımlama sadece bir taslak görevi görmektedir. Yapılan bu tanımlama sonucunda bellekte herhangi bir yer ayrılmaz. Bu yapıya ait bellekte bir yer ayrılabilmesi için en az bir yapı değişkeni tanımlanması gerekir. Geçtiğimiz bölümlerden hatırlayacağınız gibi, bu temel veri tipleri kullanılarak yapılan

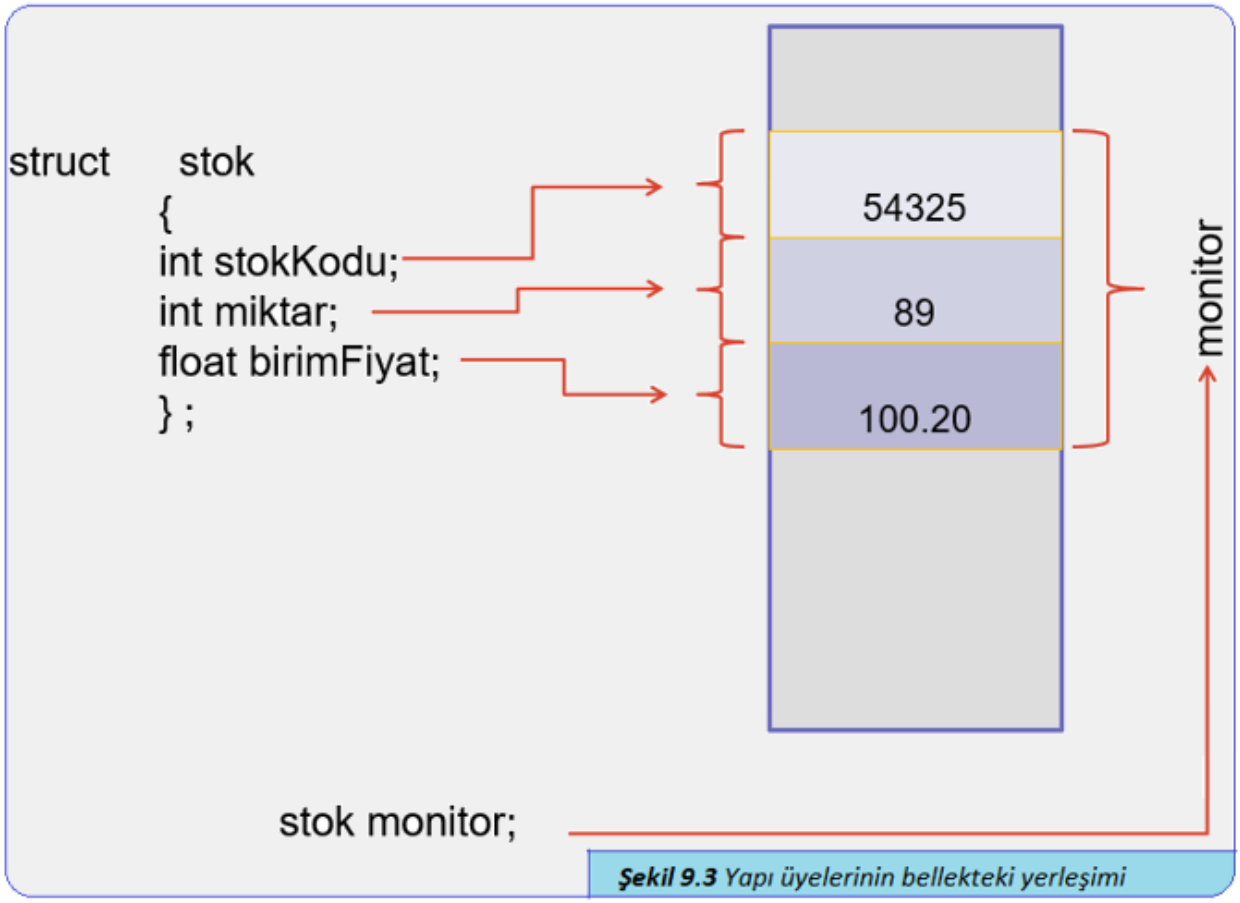
tanımlamalar için de böyleydi. Yani sadece program içerisinde ***int*** yazınca bellekte bir yer ayrılmıyordu. Bellekte yer ayrılabilmesi için, örneğin ***int sayi***; gibi bir tanımlama yapmak gerekiyordu.

Yapı değişkenlerinin tanımlanması, temel veri tipleri kullanılarak değişken tanımlanırken yapıldığı gibi, yapı ismi, boşluk, yapı değişkeninin ismi ve sonuna noktalı virgül konularak gerçekleştirilir. Aşağıda şekil 9.2'de yapı değişkeninin tanımlanması gösterilmiştir.



Şekil 9.2 *Yapı Değişkeni tanımı*

Bu tanım, bilgisayarın ana belleğinde monitör adlı yapı değişkeni için yer ayrılmasını sağlar. Ayrılan yer 12 Beyte büyüklüğündedir. Çünkü **monitor** adlı yapı değişkeninin ikisi ***int*** ve biri de ***float*** tipinde üç üyesi (***stokKodu***, ***miktar***, ***birimFiyat***) vardır ve bunlar bellekte dörder Byte 'tan toplam 12 Byte yer kaplar.



Şekil 9.3 *Yapı üyelerinin bellekteki yerleşimi*

Yapı değişkeninin bellekte ne kadar yer kapladığını ***sizeof()*** operatörünü kullanarak gösterebiliriz. ***sizeof()*** operatörü bir veri yapısının bellekte ne kadar yer kapladığını Byte cinsinden geri döndürür.

```cpp
#include <iostream>
#include <locale.h>
using namespace std;
struct stok
{
    int stokKodu;
    int miktar;
    float birimFiyat;
};
int main() {
    setlocale(LC_ALL,"Turkish");
    stok monitor;
    cout<<" Yapı Değişkeninin Bellekte Kapladığı Alan :"<<sizeof(monitor)<<" Byte"<<endl;
    return 0;
}
```

Program 9.1 *Yapı değişkeninin bellekte kapladığı alan*

```
 Yapı Değişkeninin Bellekte Kapladığı Alan :12 Byte

--------------------------------
Process exited after 0.1329 seconds with return value 0
Press any key to continue . . .
```

Program 9.1'in Ekran Çıktısı

Yapı değişkenlerine isim verilmesi sırasında uyulması gereken kurallar, temel veri tipleri ile tanımlanan değişkenlere isim verirken uyulması gereken kurallarla birebir aynıdır. Bir yapı ile aynı ifadenin içerisinde birden fazla yapı değişkeni tanımlanabilir.

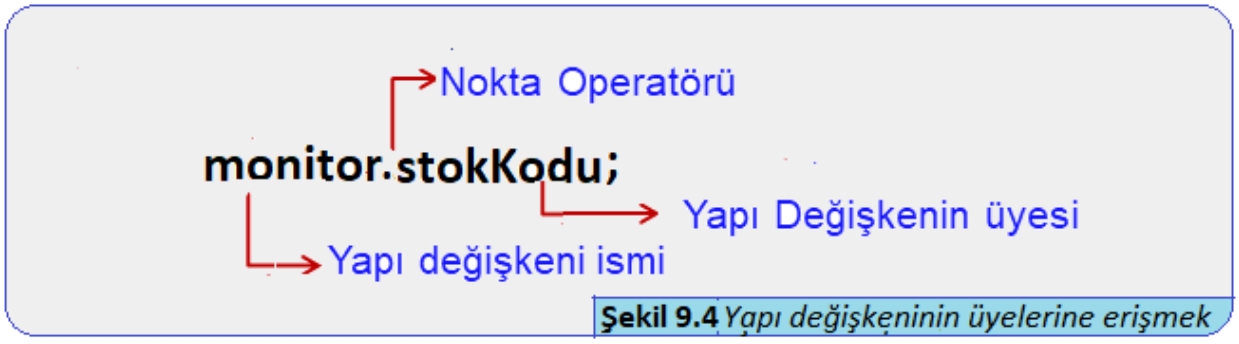
stok monitor, klavye, hardDisk, anaBellek;

Aynı ifadenin içerisinde birden fazla yapı değişkeni tanımlanması gerekiyorsa yapı değişken isimleri birbirlerinden virgül ile ayrılması gerekir. C programa dilinde yapı değişkenleri tanımlanırken tanımın başında ***struct*** bildirimi yapılması gerekir.

struct stok monitor, klavye, hardDisk, anaBellek;

C++ 'ta bu zorunluluk kaldırılmıştır. Ancak, C++ 'ta da C 'de olduğu gibi ***struct*** kullanılarak yapı değişkeni tanımlanabilir ve bu durumda da derleyici hata vermez.

Yapı değişkenlerinin üyelerine nokta operatörü(.) kullanılarak erişilir. Şekil 9.4 'te yapı değişkenin üyelerine erişilirken nokta operatörünün(üye erişim operatörü) nasıl kullanılacağı gösterilmiştir.



Şekil 9.4 *Yapı değişkeninin üyelerine erişmek*

Yapı değişkenlerinin üyelerine erişilirken sırası ile yapı değişkeninin ismi, nokta operatörü, yapı değişkeninin üyesi yazılarak erişilir.

```
stok monitor1,monitor2,monitor3;

monitor1.stokKodu=1001;
monitor1.miktar=10;
monitor1.birimFiyat=2500;
```

Yukarıda üç yapı değişkeni (***monitor1, monitor2, monitor3***) tanımlanmıştır. Daha sonra ***monitor1*** yapı değişkeninin üyelerine ilk değer ataması yapılmıştır.

Program 9.2, yapı değişkeninin üyelerine ilk değer atama işleminin nasıl yapılacağını göstermek için yazılmıştır.

```
#include <iostream>
#include <locale.h>
using namespace std;
struct stok
{
    int stokKodu;
    int miktar;
    float birimFiyat;
};
int main() {
    setlocale(LC_ALL,"Turkish");

    stok monitor1,monitor2,monitor3; //üç yapı değişkeni tanımlandı

    monitor1.stokKodu=1001;     //  monitor1 yapı değişkeninin
    monitor1.miktar=10;          //  üyelerine
    monitor1.birimFiyat=2500;   //  ilk değer ataması yapıldı
    cout<<"Stok Kodu    :"<<monitor1.stokKodu<<endl;   //monitor1 Yapı değişkeni
    cout<<"Stok Miktarı :"<<monitor1.miktar<<endl;     //üyeleri ekranda
    cout<<"Birim Fiyatı :"<<monitor1.birimFiyat<<endl; //görüntülendi

    monitor2=monitor1;      //atama operatörü kullanılarak monitör1, monitor2 'ye atandı
    cout<<endl<<endl<<"Stok Kodu    :"<<monitor2.stokKodu<<endl; //monitor2 Yapı değişkeni
    cout<<"Stok Miktarı :"<<monitor2.miktar<<endl;               //üyeleri ekranda
    cout<<"Birim Fiyatı :"<<monitor2.birimFiyat<<endl;           //görüntülendi

    monitor3={1002,25,2250}; //monitor3 yapı değişkeninin üyelerine küme parantezi ilk değer atandı.

    cout<<endl<<endl<<"Stok Kodu    :"<<monitor3.stokKodu<<endl;  //monitor3 Yapı değişkeni
    cout<<"Stok Miktarı :"<<monitor3.miktar<<endl;                //üyeleri ekranda
    cout<<"Birim Fiyatı :"<<monitor3.birimFiyat<<endl;            //üyeleri ekranda
    return 0;
}
```

Program 9.2 *Yapı değişkeninin üyelerine ilk değer atanması*

```
Stok Kodu    :1001
Stok Miktarı :10
Birim Fiyatı :2500


Stok Kodu    :1001
Stok Miktarı :10
Birim Fiyatı :2500


Stok Kodu    :1002
Stok Miktarı :25
Birim Fiyatı :2250

--------------------------------
Process exited after 0.1006 seconds with return value 0
Press any key to continue . . .
```

Program 9.2'nin Ekran Çıktısı

Program 9.2'den görülebileceği gibi yapı değişkenlerinin üyelerine tek tek değer atanabileceği gibi bir yapı değişkeni atama operatörü kullanılarak doğrudan diğer yapı değişkenine atanabilir.

monitor2=monitor1;

Bu atamadan sonra monitor2'nin üyelerinin değerleri monitor1'in üyeleri ile aynı olur. *Bakınız: Program 9.2*

Bundan başka yapı değişkenlerinin üyelerine dizilerde olduğu gibi küme parantezi içinden de atama yapılabilir.

monitor2={1002,25,2250};

Yapı değişkenlerinin üyelerine küme parantezi içerisinden atama yapılırken üyelerin değerleri arasına virgül konulması unutulmamalıdır.

Örnek: Bir yapının üyeleri öğrenci numarası, iki kısa sınav notu, vize notu, final notu ve yılsonu ortalaması olsun. Kısa sınavların her birinin yıl içi notuna katkısı %10, vizenin yıl içi notuna katkısı %80 olsun. Toplam yıl içi notunun yılsonu ortalamasına katkısı %40 ve finalin yılsonu ortalamasına katkısı da %60 olsun. ***ogrenci*** isimli bu yapıya ait ***ogrenci1*** isimli bir yapı değişkeni tanımlayınız ve yapı değişkeninin üyelerine ilk değer atayınız. Yukarıda belirtilen ağırlıkları dikkate alarak öğrencinin yılsonu ortalamasını hesaplayıp sonucu ekrana yazdıran C++ programı yazınız?

Çözüm:

```cpp
#include<iostream>
using namespace std;
struct ogrenci
{
    int ogr_no;
    int q_1;
    int q_2;
    int vize;
    int final;
    float ortalama;
};
int main()
{
    ogrenci ogrencil;
    ogrencil.ogr_no=1;
    ogrencil.q_1=70;
    ogrencil.q_2=65;
    ogrencil.vize=80;
    ogrencil.final=90;
    ogrencil.ortalama=((ogrencil.q_1*0.1)+(ogrencil.q_2*0.1)+(ogrencil.vize*0.8))*0.4+
                      ogrencil.final*0.6;
    cout<<"Ogrenci No...........: "<<ogrencil.ogr_nc<<endl;
    cout<<"Ogrnci  Not Ort........"<<ogrencil.ortalama<<endl<<endl;
return 0; }
```

Program 9.3 Öğrencinin not ortalamasını hesplayan program

```
Ogrenci No...........: 1
Ogrnci  Not Ort........85


--------------------------------
Process exited after 0.09024 seconds with return value 0
Press any key to continue . . .
```

Program 9.3'ün Ekran Çıktısı

Örnek: Bir kenarı metre ve santimetre olarak girilen karenin çevresini hesaplayan C++ programını yazınız?

Not: *kare isimli bir yapı oluşturunuz. Yapının üyeleri metre ve cm olsun. Daha sonra bu yapıya ait yapı değişkenleri tanımlayınız ve yapı değişkenlerinin üyelerinin değerlerini klavyeden giriniz.*

Çözüm:

```cpp
#include<iostream>
#include<locale.h>
using namespace std;
struct kare
{
int metre;
float cm;
};
int main()
{
    setlocale(LC_ALL,"Turkish");
    kare kenar,cevre;
    cout<<" metre Giriniz....: "; cin>>kenar.metre;
    cout<<" cm Giriniz .....: "; cin>>kenar.cm;
    cevre.cm=0;
    cevre.cm=kenar.cm*4;
    cevre.metre=0;
    if(cevre.cm>=100.0)
    {
        int ekmetre=cevre.cm/100;
        cevre.cm=cevre.cm-(ekmetre*100.0);
        cevre.metre+=ekmetre;
    }
    cevre.metre+=kenar.metre*4;
    cout<<" Karenin Çevresi ="<<cevre.metre<<" m. "<<cevre.cm<<" cm."<<endl;
    return 0;
}
```

Program 9.4 *Bir kenarı metre ve santimetre olarak ayrı girilen Karanin çevre hesabı*

```
 metre Giriniz....: 4
 cm Giriniz .....: 90.3
 Karenin Çevresi =19 m. 61.2 cm.

--------------------------------
Process exited after 8.392 seconds with return value 0
Press any key to continue
```

Program 9.4'ün Ekran Çıktısı

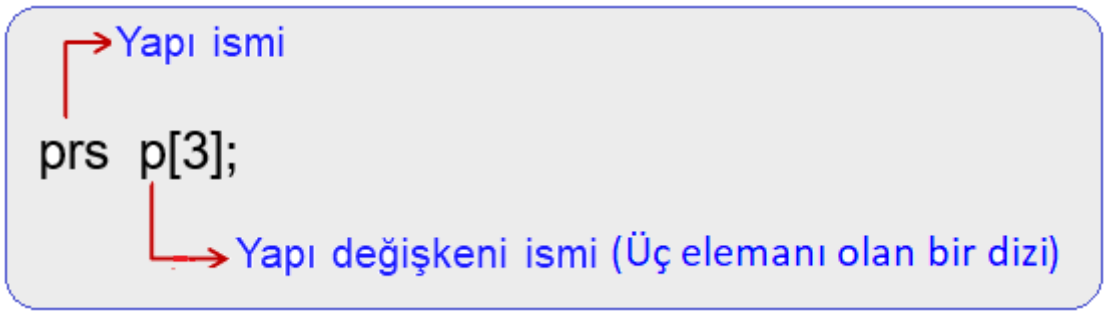
9.2. Yapıların Dizilerle Kullanılması

Altıncı bölümde C++'ın desteklediği temel veri tipleri kullanılarak bir dizinin nasıl tanımlanacağı ve C++ programları içerisinde bu dizilerin nasıl kullanılacağını öğrenmiştik. Temel veri tiplerini kullanarak tanımladığımız, kullandığımız dizilere benzer şekilde yapılarla oluşturduğumuz kendi veri türlerimizi kullanarak da dizi tanımlayabilir ve C++ programlarında tanımladığımız bu dizileri kullanabiliriz.

Aşağıda verildiği gibi bir yapı tanımlandığını varsayalım.

```
struct prs
{
    string tc;
    string ad;
    float  saatUcreti;
};
```

Bu yapıya ait bir diziyi aşağıda gösterildiği gibi tanımlayabiliriz. Bu tanımlama, ***p*** isminde 3 elemanlı ve elemanlarının türü ***prs*** olan bir diziyi belirtir.



Aşağıda verilen ***Program 9.5*** yapıların dizilerle nasıl kullanılacağını gösteren basit, öğretici bir örnektir. Program 9.5 te önce ***prs*** isminde bir tür oluşturulmuş ve daha sonra türü ***prs*** olan üç elemanlı ***p*** dizisi oluşturulmuştur.

p dizisinin elemanlarına nokta operatörü (.) kullanılarak aşağıdaki şekilde ulaşılabilir.

```
p[0].tc="987654321";
p[0].ad="Hasan Ak   ";
p[0].saatUcreti=325.5;
p[1].tc="123456789";
p[1].ad="Mehmet Kara";
p[1].saatUcreti=355.75;
p[2].tc="432198765";
p[2].ad="Fatma Pembe";
p[2].saatUcreti=300.65;
```

Program 9.5 yukarıdaki bilgiler kullanılarak aşağıdaki gibi yazılmıştır. Bu programda dizi şeklinde tanımlanmış yapı değişkeninin elemanlarına ilk değer ataması yapılmıştır. Bölüm 6'dan öğrendiğimiz şekilde, dizi şeklinde tanımlanmış yapı değişkenlerinin elemanlarını, temel veri tipleri ile tanımlanmış dizilerde olduğu gibi ***cin*** nesnesini ve çıkarma operatörünü(>>) kullanarak klavyeden de girebiliriz.

Program 9.5'in son bölümünde ***for döngüsü*** kullanılarak ***p*** dizisinin elemanları ekranda gösterilmiştir. Bu işlemin sonucu için program 9.5'in ekran çıktısını inceleyiniz.

```cpp
#include <iostream>
#include <string>
using namespace std;
struct prs
{
    string tc;
    string ad;
    float  saatUcreti;
};
int main() {
 prs p[3];
 p[0].tc="987654321";
 p[0].ad="Hasan Ak   ";
 p[0].saatUcreti=325.5;
 p[1].tc="123456789";
 p[1].ad="Mehmet Kara";
 p[1].saatUcreti=355.75;
 p[2].tc="432198765";
 p[2].ad="Fatma Pembe";
 p[2].saatUcreti=300.65;
 cout<<"TC        "<<"AD SOYAD    "<<"SAAT UCRETI"<<endl;
 cout<<"----------------------------------------"<<endl<<endl;
for(int i=0;i<3;i++) {
    cout<<p[i].tc<<" "<<p[i].ad<<" "<<p[i].saatUcreti<<endl;
}
    return 0;
}
```

Program 9.5 *Yapıların Dizilerle Kullanılması*

```
TC         AD SOYAD    SAAT UCRETI
----------------------------------------

987654321 Hasan Ak    325.5
123456789 Mehmet Kara 355.75
432198765 Fatma Pembe 300.65

--------------------------------
Process exited after 0.04084 seconds with return value 0
Press any key to continue . . .
```

Program 9.5'in Ekran Çıktısı

9.3. Yapıların Fonksiyonlara Aktarılması

Fonksiyonların parametre olarak aldığı değerler ve/veya geri dönüş değerleri yapı olabilir. Yapıları fonksiyonlara aktarmak diğer temel veri tipleri ile tanımlanmış değişkenleri fonksiyonlara aktarmaya benzer.

Program 9.6'da ***void yazdir(ogrenci);*** ifadesi ile ***yazdir*** isimli bir fonksiyon deklare edilmiştir. Bu deklarasyondan fonksiyonun bir yapı değişkenini parametre olarak alacağı anlaşılmaktadır. Ayrıca, fonksiyonun tipi **void** olarak belirlendiği için geri dönüş değeri olmayacaktır.

Daha sonra main() fonksiyonu içerisinde ***ogrenci ogrenci1;*** ifadesi ile ogrenci1 isimli bir yapı değişkeni tanımlanmış ve bu yapı değişkeninin üyelerine değerler atanmış, ortalama isimli üyenin değeri de

hesaplanarak bulunmuştur.

Programın devamında ***yazdir(ogrenci1);*** ifadesi ile fonksiyon çağrılmıştır. Çağrılan fonksiyon oldukça basittir. Bir geri dönüş değeri yoktur. Sadece kendisine gönderilen yapı değişkeninin üyelerinden ikisini (ogrencif.ogr_no, ogrencif.ortalama) ekrana yazdırmaktadır. Program 9.6'yı ve ekran çıktısını inceleyiniz.

```
#include<iostream>
using namespace std;
struct ogrenci
{
    int ogr_no;
    int q_1;
    int q_2;
    int vize;
    int final;
    float ortalama;
};
void yazdir(ogrenci);
int main()
{
    ogrenci ogrenci1;
    ogrenci1.ogr_no=1;
    ogrenci1.q_1=70;
    ogrenci1.q_2=65;
    ogrenci1.vize=80;
    ogrenci1.final=90;
    ogrenci1.ortalama=((ogrenci1.q_1*0.1)+(ogrenci1.q_2*0.1)+(ogrenci1.vize*0.8))*0.4+
                      ogrenci1.final*0.6;
    yazdir(ogrenci1);
return 0; }

void yazdir(ogrenci ogrencif)
{
    cout<<endl<<" Ogrenci No..........: "<<ogrencif.ogr_no<<endl;
    cout<<" Ogrnci  Not Ort........"<<ogrencif.ortalama<<endl<<endl;
}
```

Program 9.6 *Yapıların fonksiyonlara aktarılması*

```
 Ogrenci No..........: 1
 Ogrnci  Not Ort........85


--------------------------------
Process exited after 0.09545 seconds with return value 0
Press any key to continue . . .
```

Program 9.6'nın ekran çıktısı

Program 9.7, program 9.6'dan bir miktar farklıdır. Program 9.7 de 9.6 gibi yapıların fonksiyonlarla kullanılmasına örnek olması amacıyla yazılmıştır. Hatta 9.7'nin fonksiyonlarından biri (***yazdir***) işlev olarak 9.6'nın ***yazdır*** fonksiyonuna benzemektedir.

9.7'deki farklılık, programdaki ***hesapla*** fonksiyonunun bir yapı değişkenini çağıran fonksiyona geri döndürmesidir.

Fonksiyonlar konusunu ele aldığımız 7. Bölümden hatırlayabileceğiniz gibi, orada (7.3) "Bir fonksiyon, çalışmasını tamamladıktan sonra, kendisini çağıran programa tek bir değer döndürebilir." Şeklinde bir ifade kullanmıştık. Yapı değişkenlerini, çağıran fonksiyona geri döndürdüğümüzde de bu şekilde bir ifade kullanabiliriz. Ancak burada küçük bir fark bulunmaktadır. Çağıran fonksiyona yapı değişkeni tek olarak döndürülmüş olsa da, geri dönen yapı değişkenine, sahip olduğu birden fazla üye değişken açısından bakıldığında geri dönen değerin birden fazla olduğunu söylemek yanlış olmaz. Program 9.7'yi ve aynı programın ekran çıktısını inceleyiniz.

```
#include <iostream>
using namespace std;
struct tarih
{
    int gun;
    int ay;
    int yil;
};
tarih hesapla(tarih,tarih);
void yazdir(tarih);

int main() {
    tarih dgunu, bgun, sonuc;
    cout<<"Dogum gununu Gun  : ";cin>>dgunu.gun;
    cout<<"Dogum gununu Ay   : ";cin>>dgunu.ay;
    cout<<"Dogum gununu Yil  : ";cin>>dgunu.yil;cout<<endl;
    cout<<"Bugunun Tarihi Gun: ";cin>>bgun.gun;
    cout<<"Bugunun Tarihi Ay : ";cin>>bgun.ay;
    cout<<"Bugunun Tarihi Yil: ";cin>>bgun.yil;cout<<endl;
    sonuc=hesapla(dgunu,bgun);
    yazdir(sonuc);
    return 0;
}
tarih hesapla(tarih d,tarih b)
{
    tarih s={0,0,0};
    if(d.gun>b.gun)
    {
        b.gun+=30;
        b.ay-=1;
        }
    if(d.ay>b.ay)
    {
        b.ay+=12;
        b.yil-=1;
    }
    s.gun=b.gun-d.gun; s.ay=b.ay-d.ay;s.yil=b.yil-d.yil;
    return s;
}
void yazdir(tarih sy)
{
    cout<<"Yasiniz :"<<sy.yil<<" "<<sy.ay<<" "<<sy.gun<<endl;
}
```

Program 9.7 *Yapı Değişkenlerinin çağıran fonksiyona geri döndürülmesi*

```
Dogum gununu Gun  : 28
Dogum gununu Ay   : 05
Dogum gununu Yil  : 1999

Bugunun Tarihi Gun: 14
Bugunun Tarihi Ay : 04
Bugunun Tarihi Yil: 2020

Yasiniz :20 10 16

--------------------------------
Process exited after 30.53 seconds with return value 0
Press any key to continue . . .
```

Program 9.7'nin Ekran Çıktısı

9.4. İç İçe(Nested) Yapılar

C++'ta bir yapı içerisinde başka bir yapı bulunabilir. Program 9.8'de iki adet yapı görünüyor. Birinci yapı ***tarih*** ikinci yapı ise ***personel***. Personel içerisinde, tarih yapı türünde bir eleman bulunmaktadır.

Şimdi iç içe yapıların program içerisinde nasıl kullanılacağını açıklayalım. main() fonksiyonu içerisinde önce personel tipinde personel1 yapı değişkeni tanımlanmıştır. Sonra sırası ile aşağıda gösterildiği gibi personel1 yapı değişkeninin üyelerinin girişi yapılmıştır.

```
cout<<"personel Adi giriniz      :";
cin>>personel1.ad;
cout<<"personel Dog_Gun giriniz :";
cin>>personel1.dogtar.gun;
cout<<"personel Dog_Ay giriniz  :";
cin>>personel1.dogtar.ay;
cout<<"personel Dog_Yil giriniz :";
cin>>personel1.dogtar.yil;
```

Burada bu başlık altında şimdiye kadar öğrendiklerimizden farklı olan, personel1 yapı değişkeninin doğum tarihi (dogtar) elemanın ele alınış şeklidir. Dikkat edilirse dogtar yapı değişkeninin elemanlarına ulaşmak için bu defa iki nokta operatörü kullanılmıştır.

İpucu: *Bir yapı içerisinde kendisinden daha önce tanımlanmış bir yapı kullanılarak, eleman tanımlanmış ise, bu yapı ile tanımlanmış yapı değişkenlerin elemanlarına iki adet nokta operatörü kullanılarak ulaşılabilir*

(1)
```
struct personel
{
   char ad[20];
   struct tarih
    {
       int gun;
       int ay;
       int yil;
            };
};
```
(1)

(2)
```
struct tarih
{
  int gun;
  int ay;
  int yil;
};

struct personel
{
  char ad[20];
  tarih dogtar;
```
(2)

Yapılar iç içe kullanılırken bir yapının bildirimi doğrudan diğer yapı içinde yapılabilir**(1).**
Bu durumda yapı değişkeninin üyelerine ulaşmak için bir adet nokta operatörü kullanılır.
Önceden tanımlanmış yapı sonradan tanımlanmış yapının içerisinde kullanılabilir**(2)**

```
1   #include <iostream>
2   using namespace std;
3   struct tarih
4   {
5         int gun;
6         int ay;
7         int yil;
8         };
9   struct personel
10  {
11        char ad[20];
12        tarih dogtar;
13        };
14    int main(){
15     personel personel1;
16     cout<<"personel Adi giriniz      :";
17     cin>>personel1.ad;
18     cout<<"personel Dog_Gun giriniz :";
19     cin>>personel1.dogtar.gun;
20     cout<<"personel Dog_Ay giriniz  :";
21     cin>>personel1.dogtar.ay;
22     cout<<"personel Dog_Yil giriniz :";
23     cin>>personel1.dogtar.yil;
24     cout<<endl<<endl<<"Personel Adi : "<<personel1.ac<<endl;
25     cout<<"Dogum Tarihi : "<<personel1.dogtar.gur<<"\\"<<personel1.dogtar.ay<<"\\"<<personel1.dogtar.yil<<endl;
26     return 0;
27  }
```

Program 9.8 *İç içe yapı örneği*

```
personel Adi giriniz      :Mehmet
personel Dog_Gun giriniz :28
personel Dog_Ay giriniz  :04
personel Dog_Yil giriniz :1999


Personel Adi : Mehmet
Dogum Tarihi : 28\4\1999

--------------------------------
Process exited after 22.07 seconds with return value 0
Press any key to continue . . .
```

Program 9.8'in ekran çıktısı

Bölüm Özeti

C++'ta yapı oluşturmayı ve bu şekilde temel veri tiplerini kullanarak kendimize ait veri tipi tanımlamayı öğrendik. Oluşturulan kendimize ait veri türleri ile değişken(yapı değişkeni) tanımlamayı, yapı değişkenlerinin üyelerine nasıl ulaşacağımızı öğrendik. Yapıları kullanarak C++'ta program yazabilme becerisi kazandık. Yapı değişkenlerinin bellekte nasıl yerleştiğini ve ne kadar yer tuttuğunu öğrendik

Yapı türünde bir dizi tanımlamayı, bu dizilerin elemanlarına nasıl erişileceğini, erişilen dizi elemanlarına ilk değer atamasının nasıl yapılacağını öğrendik.

Yapıları kullanarak fonksiyon deklare etmeyi, fonksiyon tanımlamayı, fonksiyon çağırmayı, fonksiyonlardan yapı değişkenlerini döndürmeyi öğrendik.

C++ yapılar iç içe tanımlanabilir. İç içe yapı tanımlamanın iki farklı yöntemini, iç içe tanımlanmış yapıların üyelerine erişmeyi öğrendik.

Yapılar bir yönüyle sınıf ve nesnelere benzemektedir. Bu konuyu öğrenmemizin önemli yararlarından biri de sınıf ve nesneler konusunu öğrenirken bize önemli ölçüde yarar sağlayacak olmasıdır.

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