BLM267

Bölüm 5: Yapılar

C Kullanarak Veri Yapıları, İkinci Baskı
Reema Thareja

- giriiş
- İç içe yapılar
- Yapı Dizileri
- Yapılar ve Fonksiyonlar
- Öz-referanslı Yapılar

giriiş

- Bir yapı birçok bakımdan bir kayda benzer.
- Bir varlıkla ilgili ilişkili bilgileri saklar.
- Yapı, temel olarak, ilişkili bilgileri (hatta farklı veri türlerini bile) bir arada saklayabilen, kullanıcı tarafından tanımlanan bir veri türüdür.
- Bir yapı ile bir dizi arasındaki en büyük fark, bir dizinin yalnızca aynı veri türündeki bilgileri depolayabilmesidir.
- Dolayısıyla yapı, tek bir isim altında toplanan değişkenlerin bir koleksiyonudur.
- Bir yapı içerisindeki değişkenler farklı veri tiplerindendir ve her birinin yapıdan seçilmesinde kullanılan bir adı vardır.

giriiş

- Yapı Beyanı
- Bir yapı, struct anahtar sözcüğünü yapı adı takip ederek bildirilir.
- Yapının tüm değişkenleri yapı içerisinde tanımlanır.
- Bir yapı türü genellikle aşağıdaki sözdizimini kullanarak bildirilir:

```
yapı yapı–adı {
veri_türü değişken–adı;
veri_türü değişken–adı;
..... };
```

- Örneğin, bir öğrenci için bir yapı tanımlamamız gerekiyorsa, öğrenciye ilişkin ilgili bilgiler muhtemelen şunlar olacaktır: sıra numarası, ad, ders ve ücretler.
- Bu yapı şu şekilde bildirilebilir:

```
yapı öğrenci {
   int r_no;
   karakter adı[20];
   karakter kursu[20];
   yüzdürme ücretleri;
};
```

giriiş

- Artık yapı kullanıcı tanımlı bir veri tipi haline geldi.
- Bir yapı içerisinde tanımlanan her değişken ismine yapının bir üyesi denir.
- Ancak yapı bildirimi herhangi bir bellek ayırmaz veya depolama alanı tüketmez.
- Bu sadece C derleyicisine yapının bellekte nasıl düzenleneceğini ileten bir şablon verir ve ayrıca üye isimlerinin ayrıntılarını verir.
- Diğer veri tiplerinde olduğu gibi yapıya ait bir değişken tanımladığımızda yapıya bellek tahsis edilir.
- Örneğin, student değişkenini şu şekilde yazarak tanımlayabiliriz: struct student stud1;
- Burada struct student bir veri tipidir ve stud1 ise bir değişkendir.
- Değişkenleri tanımlamanın başka bir yoluna bakalım.
- Aşağıdaki sözdiziminde değişkenler yapı bildirimi sırasında bildirilir.

giriiş

```
yapı öğrenci {
int r_no;
karakter adı[20];
karakter kursu[20];
yüzdürme ücretleri;
} saplama1, saplama2;
```

- Bu bildirimde student yapısına ait stud1 ve stud2 isimli iki değişkeni bildiriyoruz.
- Yani yapının birden fazla değişkenini bildirmek istiyorsanız değişkenleri virgül kullanarak ayırın.
- Yapının değişkenlerini bildirdiğimizde, her değişken için ayrı bellek tahsis edilir. Bu Şekil 5.1'de gösterilmiştir.

giriiş

- Son olarak, yapı üye adları ve yapı adları, sıradan değişken adları için belirlenen kurallara uymaktadır.
- Ancak yapı adı ile yapı elemanı adının aynı olmamasına dikkat edilmelidir.
- Ayrıca yapı adı ile değişken adının da farklı olması gerekir.

giriiş

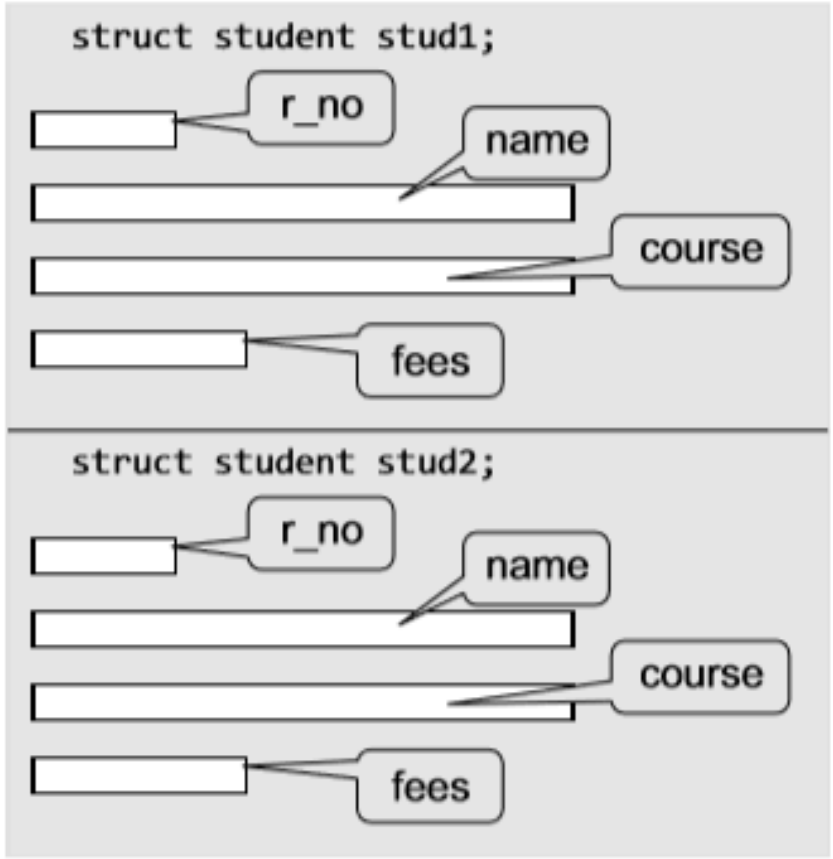


Figure 5.1 Memory allocation for a structure variable

giriiş

- Typedef Bildirimleri
- Typedef (tip tanımından türetilir) anahtar kelimesi programcının var olan bir veri tipini kullanarak yeni bir veri tipi adı oluşturmasını sağlar.
- typedef kullanıldığında yeni veri oluşturulmaz, bunun yerine bilinen bir veri türüne alternatif bir isim verilir.
- Typedef anahtar sözcüğünün genel sözdizimi şu şekilde verilir: typedef existing_data_type new_data_type;
- typedef ifadesinin herhangi bir bellek kaplamadığını, sadece yeni bir tip tanımladığını unutmayın.
- Örneğin typedef int INTEGER yazarsak; INTEGER, int veri tipinin yeni ismi olacaktır.
- Değişkenleri yeni veri türü adını kullanarak bildirmek için, değişken adından önce veri türü adını (yeni) ekleyin.
- Dolayısıyla, bir tamsayı değişkeni tanımlamak için şimdi INTEGER num=5 yazabiliriz;

giriiş

- Bir yapı adının önüne typedef anahtar sözcüğünü eklediğimizde yapı yeni bir türe dönüşür.
- C tarafından önceden tanımlanmış tipler veya sizin tanımladığınız tipler için yapıyı daha anlamlı isimlerle daha kısa hale getirmek için kullanılır.
- Örneğin, aşağıdaki bildirimi ele alalım:

```
typedef yapı öğrenci
{
   int r_no;
   karakter adı[20];
   karakter kursu[20];
   yüzdürme ücretleri;
}Öğrenci;
```

- Artık yapının adının önüne typedef anahtar sözcüğünü eklediğimize göre, Studen yeni bir veri türü haline gelir.
- Dolayısıyla, artık int, float, char, double, vb. türündeki değişkenleri bildirdiğiniz gibi, bu yeni veri türünün değişkenlerini de doğrudan bildirebilirsiniz.
- student yapısındaki bir değişkeni bildirmek için şunu yazabilirsiniz:

```
Öğrenci öğrenci1;
```

- Dikkat ederseniz struct student stud1'i yazmadık.

giriiş

- Yapıların Başlatılması
- Bir yapı, diğer veri tiplerinin başlatıldığı şekilde başlatılabilir.
- Bir yapının başlatılması, yapının üyelerine bazı sabitlerin atanması anlamına gelir.
- Kullanıcı yapıyı açıkça başlatmadığında, C bunu otomatik olarak yapar.
- int ve float üyeleri için değerler sıfır olarak başlatılır ve char ve string üyeleri varsayılan olarak '\0' olarak başlatılır.
- Başlatıcılar parantez içine alınır ve virgülle ayrılır.
- Ancak başlatıcıların yapı tanımındaki karşılık gelen tipleriyle eşleşmesine dikkat edilmelidir.

giriiş

- Bir yapı değişkenini başlatmanın genel sözdizimi aşağıdaki gibidir:

```
yapı yapı_adı
{
   veri_türü üye_adı1;
   veri_türü üye_adı2;
   veri_türü üye_adı3;
   ..... }struct_var = {sabit1, sabit2, sabit3,...};
yapı yapı_adı
{
   veri_türü üye_adı1;
   veri_türü üye_adı2;
   veri_türü üye_adı3;
   .....
};
```

- C, typedef tanımı oluşturulurken değişken bildirimine izin vermez.
- Yani değişkenler bağımsız bir ifadede tanımlanmalıdır.

```
yapı yapı_adı yapı_değişkeni = {sabit1, sabit2, sabit 3,...};
```

giriiş

- Örneğin, bir öğrenci yapısını yazarak başlatabiliriz,

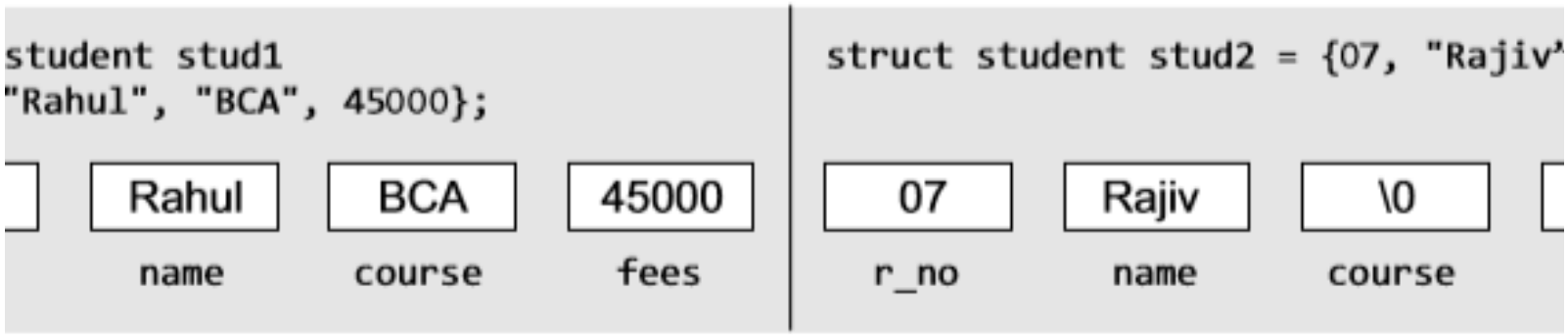
```
yapı öğrenci {
    int r_no;
    karakter adı[20];
    char kursu[20]; yüzme ücretleri;
}stud1={01,"Rahul","BCA",45000};
```

- Veya yazarak,

```
struct öğrenci stud1 ={01,"Rahul","BCA",45000};
```

- Şekil 5.2, değerlerin yapının bireysel alanlarına nasıl atanacağını göstermektedir.
- Bir yapının tüm üyeleri başlatılmadığında buna kısmi başlatma denir.
- Kısmi başlatma durumunda yapının ilk birkaç üyesi başlatılır ve başlatılmayanlara varsayılan değerler atanır.

giriiş



Assigning values to structure elements

giriiş

- Bir Yapının Üyelerine Erişim
- Bir yapının her bir üyesi tıpkı normal bir değişken gibi kullanılabilir, ancak ismi biraz daha uzun olacaktır.
- Bir yapı üyesi değişkenine genellikle '.' (nokta) operatörü kullanılarak erişilir.
- Bir yapıya veya yapının bir üyesine erişim sözdizimi şu şekilde verilebilir: struct_var.member_name
- Nokta operatörü yapının belirli bir üyesini seçmek için kullanılır.
- Örneğin, stud1 yapı değişkeninin bireysel veri üyelerine değerler atamak için şunu yazabiliriz:

```
stud1.id =1;
stud1.name = "Rahul";
stud1.sınıf = "BCA";
stud1.ücretleri = 45000;
```

- stud1 yapı değişkeninin veri üyeleri için değerleri girmek için şunu yazabiliriz:
- scanf("%d", &stud1.r_no);
- scanf("%s", stud1.name);

giriiş

- Benzer şekilde, stud1 yapı değişkeninin değerlerini yazdırmak için şunu yazabiliriz:

```
printf("%s",stud1.sınıf);
printf("%f",stud1.ücretler);
```

- Sadece yapının değişkenlerini bildirdiğimizde bellek tahsis edilir.
- Başka bir deyişle, bellek yalnızca yapıyı örneklendirdiğimizde tahsis edilir.
- Herhangi bir değişkenin yokluğunda yapı tanımı, struct tipinde bir değişken tanımlandığında bellekte yer ayırmak için kullanılacak bir şablondan ibarettir.
- Bir yapının değişkenleri tanımlandıktan sonra bunlar üzerinde birkaç işlem yapabiliriz.
- Örneğin, bir değişkenin değerlerini başka bir değişkene atamak için atama operatörünü (=) kullanabiliriz.

giriiş

- Yapıları Kopyalama ve Karşılaştırma
- Bir yapıyı aynı tipteki başka bir yapıya atayabiliriz.
- Örneğin, struct student tipinde stud1 ve stud2 adında iki yapı değişkenimiz varsa, aşağıdaki gibi verilir:

  struct öğrenci stud1={01,"Rahul", "BCA" 45000};

  yapı öğrenci stud2;
- Daha sonra bir yapı değişkenini diğerine atamak için şunu yazacağız:

  saplama2 = saplama1;
- Bu ifade stud2'nin üyelerini stud1'in üyelerinin değerleriyle başlatır.
- Dolayısıyla artık stud1 ve stud2 değerleri Şekil 5.3'te görüldüğü gibi verilebilir.

giriiş

```
struct student stud1
= {01, "Rahul", "BCA", 45000};
```

01	Rahul	BCA	45000
r_no	name	course	fees

```
struct student stud2 = stud1;
```

01	Rahul	BCA	45000
r_no	name	course	fees

Figure 5.3 Values of structure variables

giriiş

- C, bir yapı değişkeninin bir diğeriyle karşılaştırılmasına izin vermez.
- Ancak bir yapının bireysel üyeleri, başka bir yapının bireysel üyeleriyle karşılaştırılabilir.
- Bir yapı üyesini başka bir yapının üyesiyle karşılaştırdığımızda, karşılaştırma herhangi bir sıradan değişken karşılaştırması gibi davranacaktır.
- Örneğin, iki öğrencinin ücretlerini karşılaştırmak için if(stud1.fees > stud2.fees) yazacağız //stud1'in ücretinin stud2'den büyük olup olmadığını kontrol etmek için

İç içe yapılar

- Bir yapı başka bir yapının içerisine yerleştirilebilir, yani bir yapı başka bir yapıyı üyesi olarak içerebilir.
- Bir başka yapıyı üyesi olarak barındıran yapıya iç içe yapı denir.
- Şimdi iç içe yapıların nasıl tanımlandığını görelim.
- İç içe geçmiş bir yapıyı tek bir bildirimle bildirmek mümkün olsa da önerilmez.
- Daha kolay ve anlaşılır yol, yapıları ayrı ayrı bildirmek ve daha sonra bunları daha üst seviyedeki yapıda gruplandırmaktır.
- Bunu yaparken, iç içe yerleştirmenin içeriden dışarıya (en alt seviyeden en kapsayıcı seviyeye doğru) yapılmasına dikkat edin, yani önce en içteki yapıyı, sonra bir sonraki seviyedeki yapıyı bildirin ve en dıştaki (en kapsayıcı) yapıya doğru ilerleyin.

İç içe yapılar

```
typedef struct { char ilk_ad[20];
   char orta_ad[20];
   char soyadı[20];
}İSİM;
typedef struct { int gg; int aa; int yy; }TARİH;
typedef yapı { int r_no;
   İSİM adı;
   karakter kursu[20];
   DOB VERİLERİ;
   yüzdürme ücretleri;
} öğrenci;
```

İç içe yapılar

- Bu örnekte student yapısının NAME ve DATE adında iki yapıyı daha içerdiğini görüyoruz.
- Her iki yapının da kendine ait alanları var.
- NAME yapısı üç alana sahiptir: first_name, mid_name ve last_name.
- DATE yapısı da gg, aa ve yy olmak üzere üç alana sahiptir ve bunlar tarihin gününü, ayını ve yılını belirtir.
- Şimdi yapı alanlarına değer atamak için student stud1 yazacağız;

```
saplama1.r_no=1;
stud1.name.first_name="Ana";
stud1.name.mid_name="Raj";
stud1.name.last_name="Thareja";
stud1.course="BCA";
stud1.DOB.dd=15;
saplama1.DOB.mm=09;
stud1.DOB.yy=1990;
stud1.ücretler=45000;
```

- İç içe geçmiş yapılarda, en içteki ve en dıştaki yapıların üyelerine erişmek için yapı değişkenleriyle birlikte nokta operatörünü kullanırız.

Yapı Dizileri

- Yukarıdaki örneklerde bir yapının nasıl tanımlanacağını ve veri üyelerine nasıl değer atanacağını gördük.
- Şimdi, bir yapı dizisinin nasıl tanımlandığını tartışacağız.
- Bu amaçla öncelikle dizi yapılarına nerede ihtiyaç duyacağımızı analiz edelim.
- Sınıfta tek bir öğrencimiz yoktur.
- Ama en az 30 öğrenci olabilir.
- Yani yapının aynı tanımı 30 öğrencinin tamamı için kullanılabilir.
- Bu, bir dizi yapı oluşturduğumuzda mümkün olacaktır.
- Yapıların dizisi, yerleşik veri tipindeki bir diziyi bildirdiğimiz şekilde bildirilir.

Yapı Dizileri

- Bir dizi yapının istendiği bir başka örnek ise bir organizasyondur.
- Bir örgütün çok sayıda çalışanı vardır.
- Dolayısıyla her çalışan için ayrı bir yapı tanımlamak uygulanabilir bir çözüm değildir.
- Yani burada tüm çalışanlar için ortak bir yapı tanımı yapabiliriz.
- Bu da yine employee yapısının bir dizisini tanımlayarak yapılabilir.
- Bir yapı dizisini bildirmek için genel sözdizimi şu şekilde verilebilir: struct struct_name { data_type member_name1; data_type member_name2; data_type member_name3; ..... }; struct struct_name struct_var[indeks];

Yapı Dizileri

- Verilen yapı tanımını göz önünde bulundurun.
- struct öğrenci { int r_no; char adı[20];
- char kursu[20]; yüzdürme ücretleri; };
- Bir öğrenci dizisi, struct student stud[30] yazılarak bildirilebilir;
- Şimdi, sınıftaki i'inci öğrenciye değer atamak için şunu yazacağız:
- saplama[i].r_no = 09;
- stud[i].name = "RASHI";
- stud[i].kurs = "MCA";
- stud[i].ücretler = 60000;
- Yapı değişkenleri dizisini bildirim anında başlatmak için aşağıdaki gibi yazabiliriz: struct student stud[3] = {{01, "Aman", "BCA", 45000},{02, "Aryan", "BCA", 60000}, {03, "John", "BCA", 45000}};

Yapılar ve Fonksiyonlar

- **Yapıların tam anlamıyla kullanışlı olabilmesi için onları fonksiyonlara geçirip döndürecek bir mekanizmaya sahip olmamız gerekir.**
- **Bir fonksiyon, Şekil 5.4'te gösterildiği gibi bir yapının üyelerine üç şekilde erişebilir.**

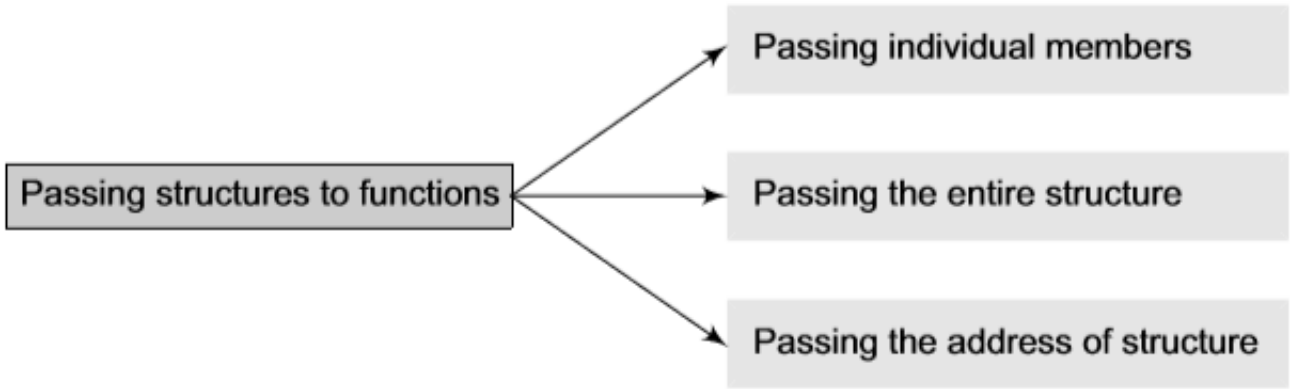


Figure 5.4 Different ways of passing structures to functions

Yapılar ve Fonksiyonlar

- **Bireysel Üyeleri Geçirme Bir yapının herhangi bir bireysel üyesini bir fonksiyona geçirmek için, bireysel üyelere başvurmak üzere doğrudan seçim operatörünü kullanmalıyız. Çağrılan program, bir değişkenin sıradan bir değişken mi yoksa yapılandırılmış bir üye mi olduğunu bilmez. Bu kavramı gösteren aşağıda verilen koda bakın.**

```
#include <stdio.h>
typedef struct
{
      int x;
      int y;
}POINT;
void display(int, int);
int main()
{
      POINT p1 = {2, 3};
      display(p1.x, p1.y);
      return 0;
}
void display(int a, int b)
{
      printf(" The coordinates of the point are: %d %d", a, b);
}
```

Output

```
The coordinates of the point are: 2 3
```

Yapılar ve Fonksiyonlar

- Tüm Yapının Geçilmesi
- Tıpkı diğer değişkenler gibi, bir yapının tamamını bir fonksiyon argümanı olarak geçirebiliriz.
- Bir yapı argüman olarak geçirildiğinde, değere göre çağrı yöntemi kullanılarak geçirilir, yani yapının her üyesinin bir kopyası yapılır.
- Bir yapıyı bir fonksiyona geçirmek ve bir yapı döndürmek için genel sözdizimi şu şekilde verilebilir: struct struct_name func_name(struct struct_name struct_var);
- Yukarıdaki söz dizimi gereksinime göre değişebilir.
- Örneğin, bazı durumlarda bir fonksiyonun bir yapı almasını ancak bir void veya başka bir veri tipinin değerini döndürmesini isteyebiliriz.
- Aşağıda verilen kod, değere göre çağırma metodunu kullanarak bir fonksiyona bir yapı geçirir.

Yapılar ve Fonksiyonlar

```
#include <stdio.h>
typedef struct
{
        int x;
        int y;

}POINT;
void display(POINT);
int main()
{
        POINT p1 = {2, 3};
        display(p1);
        return 0;
}
void display(POINT p)
{
        printf("The coordinates of the point are: %d %d", p.x, p.y);
}
```

Yapılar ve Fonksiyonlar

- Çağrılan fonksiyona bir yapı geçirirken dikkat edilmesi gereken bazı noktaları özetleyelim.
- Çağrılan fonksiyon yapının tamamının bir kopyasını döndürüyorsa struct olarak tanımlanmalı ve ardından yapı adı eklenmelidir.
- Fonksiyon bildiriminde parametre olarak kullanılan yapı değişkeni, çağrılan fonksiyondaki gerçek argümanla aynı olmalıdır (ve bu, yapı türünün adı olmalıdır)
- Bir fonksiyon bir yapı döndürdüğünde, çağıran fonksiyonda döndürülen yapının aynı tipteki bir yapı değişkenine atanması gerekir.

Yapılar ve Fonksiyonlar

- Yapıları İşaretçiler Aracılığıyla Geçirme
- Değerle çağırma yöntemini kullanarak fonksiyonlara büyük yapılar geçirmek çok verimsizdir.
- Bu nedenle yapıların işaretçiler üzerinden geçirilmesi tercih edilir.
- C'de kullanıcı tanımlı tipler de dahil olmak üzere hemen hemen her tipe işaretçi oluşturmak mümkündür.
- Yapılara işaretçiler oluşturmak son derece yaygındır.
- Diğer durumlarda olduğu gibi, bir yapıya ait işaretçi asla kendi başına bir yapı değildir, yalnızca yapının adresini tutan bir değişkendir.

Yapılar ve Fonksiyonlar

- Bir yapıya işaretçi bildirmenin sözdizimi şu şekilde verilebilir:
- yapı yapı_adı {
- veri_türü üye_adı1;
- veri_türü üye_adı2;
- veri_türü üye_adı3;
- ..... }*ptr;
- veya, struct struct_name *ptr;
- Öğrenci yapımız için struct student *ptr_stud, stud; yazarak bir işaretçi değişkeni tanımlayabiliriz.

Yapılar ve Fonksiyonlar

- Yapılacak bir sonraki şey, diğer işaretçilerde yaptığımız gibi, adres operatörünü (&) kullanarak stud'ın adresini işaretçiye atamaktır.
- Yani adresi atamak için ptr_stud = &stud; yazacağız. Bir yapının üyelerine erişmek için /* yapıyı al, sonra bir üye seç */ (*ptr_stud).roll_no; yazabiliriz.
- Parantezlerin * işaretinden daha yüksek önceliği olduğundan, bu ifadeyi yazmak işe yarayacaktır.
- Ama bu ifadeyle çalışmak, özellikle yeni başlayanlar için hiç de kolay değil.
- Yani, C aynı görevi yapmak için yeni bir operatör sunar. Bu operatör 'işaret eden' operatör (->) olarak bilinir. Şu şekilde kullanılabilir: /* ptr_stud yapısındaki roll_no işaret eder */ ptr_stud -> roll_no = 01;
- Bu ifade alternatifinden çok daha kolaydır.

Öz-referanslı Yapılar

- Kendine referanslı yapılar, aynı türdeki verilere referans içeren yapılardır.
- Yani, kendi kendine referans veren bir yapı, diğer verilerin yanı sıra, yapının verisiyle aynı tipte bir veriye işaret eden bir işaretçi de içerir.
- Örneğin aşağıda verilen yapı düğümünü ele alalım.

```
yapı düğümü
{
   tamsayı değer;
   yapı düğümü *sonraki;
};
```

Öz-referanslı Yapılar

- Burada yapı düğümü iki tür veri içerecektir: bir tamsayı val ve bir sonraki işaretçi.
- Böyle bir yapıya neden ihtiyaç duyduğumuzu merak ediyor olmalısınız.
- Aslında öz-referanslı yapı diğer veri yapılarının temelini oluşturur.
- Bunları bu kitap boyunca kullanacağız ve bağlantılı listeler, ağaçlar ve grafikler hakkında konuştuğumuzda bunların amacı sizin için daha açık hale gelecek.