Giriş

Katarlar anlaşılması en zor konulardan biridir. C programlama dilinde iki tırnak içine alınan her ifadeye katar denir. Örneğin:

```
"Izmir"
"sonuc = %d\n"
"Devam etmek icin ENTER tusuna basin."
```

Türkçe yazılan C kitaplarda, ingilizce *string* kelimesi yerine aşağıdaki ifadelerden biri karşılaşılabilir:

katar = karakter topluluğu = karakter dizisi = sözce = sicim

Anlatımda, *katar* terimini kullanacağız.

12.1 Katar Bildirimi

Katarlar, `char` tipinde bildirilen karakter dizileridir ve harfler, rakamlar, veya bazı sembolleriden oluşur. C dilinde katar bildirimi için bir tip deyimi yoktur. Bu yüzden, bir katara bir dizi veya gösterici gözüyle bakılır. Genel olarak bir katarın bildirimi:

```
char katar_adı[eleman_sayısı];
```

yada

```
char *katar_adı;
```

şeklinde yapılır. Örneğin bir öğrencinin isim bilgisi `ad` adlı bir katarla tutulmak istenirse:

```
char ad[10];
```

yada

```
char *ad;
```

şeklinde programın başında bildirilmelidir.

12.2 Katarlara Başlangıç Değeri Atama

Diğer dizi bildirimlerinde olduğu gibi, karakter dizilerine başlangıç değeri verilebilir. Örneğin aşağıda verilen iki bildirim aynı anlamdadır:

```
char s[5]={'I','z','m','i','r','\0'};
char s[5]="Izmir";
```

Birinci satırdaki bildirimde `'\0' (NULL)` sonlandırıcı karakter dizisinin sonlandığını gösterir. Daha önce de bahsedildiği gibi sonlandırıcı karakter, karakter dizileri üzerinde işlemlerin hızlı ve etkin bir biçimde yapılabilmesine olanak sağlar. İkinci bildirimde buna gerek yoktur.

Eğer bir karakter dizisinin kaç eleman sayısı belirtilmezse, başlangıçta bildirilen karakter sayısı kaç tane ise dizinin eleman sayısı o kadar olduğu varsayılır.

```
char s[] = "Ankara";     /* 6 elemanlı */
```

Eğer bildirim gösterici ile yapılmak isterirse:

```
char *s = "Ankara";     /* 6 elemanlı */
```

yada

```
char *s;
s = "Ankara";
```

Ancak

```
char s[6];
s = "Ankara";
```

şeklindeki bir atama geçersizdir. Çünkü bu şekilde yapılan bildirimde `s` bir değişken değil dizidir.

Elemanları katar olan diziler tanımlamak mümkündür. Örneğin en uzunu 7 karakter olan 5 farklı isim bir çatı altında şöyle toplanabilir:

```
char isim[5][8] = { "Semra", "Mustafa", "Ceyhun", "Asli", "Leyla" };
```

yada

```
char isim[][8]  = { "Semra", "Mustafa", "Ceyhun", "Asli", "Leyla" };
```

yada

```
char *isim[5]   = { "Semra", "Mustafa", "Ceyhun", "Asli", "Leyla" };
```

Görüldüğü gibi, bu tip tanımlamalarda birinci boyut (satır) dizinin eleman sayısını, ikinci boyut (sütun) her bir elemanın sahip olabileceği maksimum karakter sayısını gösterir.

NOT

Katar ifadelerinde doğrudan çift tırnak `"` veya ters bölü `\` karakterleri kullanılamaz. Bu durumda katar ifadeleri içerisinde

`"` yerine `\"`
`\` yerine `\\`

kullanılmalıdır. Örneğin:

```
char *mes = "\"ilk.c\" dosyasinin yeri:";
char *yol = "C:\\WINDOWS\\DESKTOP\\C";
...
puts(mes);
puts(yol);
```

ile ekrana aşağıdaki satırlar bastırlır:

```
"ilk.c" dosyasinin yeri:
C:\WINDOWS\DESKTOP\C
```

12.3 Katarlar Üzerinde İşlem Yapan Standart G/Ç Fonksiyonları

`printf()` ve `scanf()` fonksiyonları diğer tiplerde olduğu gibi formatlı okuma/yazma amaçlı kullanılır. Katar formatı `%s` dir. Örneğin:

```
char str[20];
```

```
...
scanf("%s",str);
printf("%s\n",str);
```

satırları ile klavyeden okunan katarın ilk 20 karakteri ekrana yazdırlabilir. Burada `printf()` fonksiyonu:

```
printf(str);
```

şeklinde de kullanılabir. Bu durumda, katar ekrana yazdırılır fakat imlec (cursor) bir alt satıra geçmez.

`gets()` fonksiyonu klavyeden karakter dizisi almakta kullanılan bir C fonksiyonudur. Bu fonksiyon, klavyeden girilen karakterleri diziye yerleştirdikten sonra dizinin sonuna otomatik olarak `NULL` (`'\0'`) karakterini ekler.

```
char str[20];
...
gets(str);
```

NOT

`gets()` fonksiyonunu kullanmak biraz tehlikeli olabilir. Çünkü, `gets()` ile okuma yapılırken katarın büyüklüğünü dikkate alınmaz. Örneğin:

```
char s[10];
gets(s);
```

şeklindeki okuma işleminde `s` en fazla 10 karakter saklayabilirken, `gets()` ile 100 karakter girilirse, derleyici bütün karakterleri saklamaya çalışır. Bu durumda, program sağlıklı çalışmaz ve hata verir. Bu yüzden bazı derleyiciler, `gets()` kullanıldığında aşağıdaki gibi bir uyarı verir.

```
warning: the `gets' function is dangerous and
should not be used.
```

Sonuç olarak, `scanf()` fonksiyonunu kullanmanız tavsiye edilir.

`puts()` fonksiyonu bir karakter dizisini ekrana yazdırmak için kullanılır. Bu fonksiyon diziyi ekrana yazdırdıktan sonra imleci (cursor) bir sonraki satıra geçirir.

```
char *str = "Hangi cilgin bana zincir vuracakmis sasarim";
...
puts(str);
```

`puts(str)` ile `printf("%s\n",str)` işlevsel olarak birbirine eşdeğerdir.

Buraya kadar anlatılanlar Program 12.1-3 de özetlenmiştir.

Program 12.1: *Bir katarın farklı yöntemlerle ekrana yazdırılması*

```
01: /* 12prg01.c
02:    Bir katarın farklı yöntemlerle ekrana yazılması */
03:
```

```
04: #include <stdio.h>
05:
06: int main()
07: {
08:      char dizi[7] = {'S', 'e', 'l', 'a', 'm', '!',
09: '\0'};
10:      int i;
11:
12:      /* Herbir karakteri ayrı ayrı alt alta yaz */
13:      printf("Dizi elemanlari:\n");
14:      for (i=0; i<7; i++)
15:         printf("dizi[%d] icerigi: %c\n", i,
16: dizi[i]);
17:      printf("\n");
18:
19:      /* 1. yöntem: her elemanı yanyana yaz */
20:      printf( "Butun dizi (1.yontem): ");
21:      for (i=0; i<7; i++)
22:         printf("%c", dizi[i]);
23:
24:      /* 2. Yöntem: bütün diziyi yaz */
25:      printf("\nButun dizi (2.yontem): ");
26:      printf("%s\n", dizi);
27:
28:      printf("\n");
29:
     return 0;
    }
```

ÇIKTI

```
Dizi elemanlari:
dizi[0] icerigi: S
dizi[1] icerigi: e
dizi[2] icerigi: l
dizi[3] icerigi: a
dizi[4] icerigi: m
dizi[5] icerigi: !
dizi[6] icerigi:

Butun dizi (1.yontem): Selam!
Butun dizi (2.yontem): Selam!
```

Aşağıdaki program kalvyeden girilen bir katar içindeki `'m'` karakterlerinin sayısını blup ekrana yazar. İnceleyiniz.

Program 12.2: *Bir katar içinde 'm' karakterinin sayısını öğrenme*

```
01: /* 12prg02.c: Bir stringin içindeki 'm'
02: karakterlerinin sayısı hesaplar */
03:
04: #include <stdio.h>
05:
06: int main()
07: {
08:    char str[20];
09:    int  i,sayac=0;
10:
11:    printf("Bir string girin: ");
```

```
12:     gets(str);
13:
14:     for(i=0; str[i] != '\0'; i++)
15:        if( str[i] == 'm') sayac++;
16:
17:     printf("'m' karakteri sayisi = %d\n",sayac);
18:
19:   return 0;
    }
```

ÇIKTI

```
Bir katar girin: marmara
'm' karakteri sayisi = 2
```

13. satırdaki döngüde, `str[i]!='\0'`, koşulu ile sonlandırıcı karaktere gelinip gelinmedği sorgulanmaktadır. 14. satırda katar içindeki `'m'` karakterine rastlanırsa `sayac` değeri bir artmaktadır. Katar sonuna kadar bütün `'m'` karakterlerinin toplamı hesaplanıp ekrana yazdırılmıştır.

Program 12.2'deki döngü şöyle de yazılabilirdi:

```
...
for(i=0; str[i]; i++)
   if(str[i] == 'm') sayac++;
...
```

Buradaki işlemle `str[i]`, `NULL` karakterinden farklı olduğu sürece döngü devam ettirilmiştir.

Aşağıdaki program elemanlı katar olan bir karakter dizisini ekrana yazar.

Program 12.3: *Bir kataru yazdırma*

```
01: /* 12prg03.c: Bir elemanları katar olan karakter
02: dizisini yazdırma */
03:
04: #include <stdio.h>
05:
06: int main()
07: {
08:   char *gun[7] = { "Pazartesi", "Sali", "Carsamba",
09:                    "Persembe", "Cuma",
10: "Cumartesi", "Pazar" };
11:   int i;
12:
13:   for(i=0; i<7; i++)
14:     printf("%d. %s\n",i+1,gun[i]);
15:
16:  return 0;
    }
```

ÇIKTI

```
1. Pazartesi
2. Sali
3. Carsamba
4. Persembe
5. Cuma
```

```
6. Cumartesi
7. Pazar
```

12.4 Bazı Katar Fonksiyonları

Bu fonksiyonlar standart C'de iki katarı karşılaştırmak, bir katarın içeriğini diğerine kopyalamak ve katarın uzunluğunu bulmak vb işlemler için tanımlı fonksiyonlardır. Bu ve benzeri fonksiyonlar kullanılırken `string.h` kütüphanesi programın başına ilave edilmelidir. Burada, bunlardan bir kaçı Tablo 12.1 de verilmiştir.

Tablo 12.1: `string.h` *kütüphanesine ait, bazı katar fonksiyonları*

Fonksiyon	Açıklama
`int strcmp(char *str1,char *str2);`	`str1` ve `str2` yi karşlaştırır. Eşitse 0, `str1` büyükse 0'dan büyük bir değer aksi halde 0'dan küçük bir değer göderir.
`char *strcpy(char *str1, char *str2);`	`str2` yi `str1` e kopyalar
`char *strcat(char *str1,char *str2);`	`str2` yi `str1` e ekler
`char *strrev(str);`	`str` yi ters çevirir (`NULL` karakteri hariç)
`int strlen(str);`	`str` nin kaç karakterden oluştuğunu hesaplar
`char *strchr(char *str,char kr);`	`kr` karakterinin `str` içindeki (baştan itibaren) ilk karşılaştığı yeri verir
`char *strstr(char *str1,char *str2);`	`str2` katarının `str1` içindeki (baştan itibaren) ilk karşılaştığı yeri verir
`char *strlwr(char *str);`	`str` nin bütün karakterini küçük harfe çevirir
`char *strupr(char *str);`	`str` nin bütün karakterini büyük harfe çevirir

Bu fonksiyonların kullanılması Program 12.4-8'de verilmiştir. Programları Dikkatle inceleyiniz.

Program 12.4: `strcmp` *fonksiyonunun kullanımı*

```
01: /* 12prg04.c: Basit bir şifre programı.
02:    Kullanıcı en fazla 3 kez yanlış şifre girebilir.
03: */
```

```
04:
05:  #include <stdio.h>
06:  #include <string.h>
07:
08:  int main()
09:  {
10:     char sifre[8];
11:     int  sonuc, hak=3;
12:
13:     while( hak-- > 0 )
14:     {
15:        printf("Sifre : ");
16:        gets(sifre);                    /* şifreyi al
17:  */
18:
19:        sonuc = strcmp(sifre,"elma%xj4");
20:
21:        if( sonuc==0 ){                 /* şifre
22:  kontrol */
23:           puts("sifre dogru");
24:           break;
25:        }
26:        else
27:           puts("sifre yanlis");
28:     }

      return 0;
    }
```

ÇIKTI

```
Sifre : admin
sifre yanlis
Sifre : root
sifre yanlis
Sifre : elma%xj4
sifre dogru
```

Program 12.5: `strcpy` *fonksiyonunun kullanımı*

```
01:  /* 12prg05.c: Bir katarı diğerine kopyalama */
02:
03:  #include <stdio.h>
04:  #include <string.h>
05:
06:  int main()
07:  {
08:     char str1[] = "Deneme";
09:     char str2[15], str3[15];
10:     int  i;
11:
12:     /* strcpy kullanarak  kopyalama */
13:     strcpy(str2, str1);
14:
15:     /* strcpy kullanmadan kopyalama */
```

```
16:     for(i=0; str1[i]; i++)
17:        str3[i] = str1[i];
18:     str3[i] = '\0';       /* sonlandırıcı ekle */
19:
20:     /* sonuçlar ekrana */
21:     printf("str1 : %s\n",str1);
22:     printf("str2 : %s\n",str2);
23:     printf("str3 : %s\n",str3);
24:
25:   return 0;
26: }
```

ÇIKTI

```
str1 : Deneme
str2 : Deneme
str3 : Deneme
```

Program 12.6: `strcat` *fonksiyonunun kullanımı*

```
01: /* 12prg06.c: Bir katarı diğerine ekler */
02:
03: #include <stdio.h>
04: #include <string.h>
05:
06: int main()
07: {
08:    char mesaj[20] = "Selam ";  /* 1. katar */
09:    char  isim[10];             /* 2. katar */
10:
11:    printf("Adiniz ? : ");
12:    scanf("%s",isim);
13:
14:    /* ekle */
15:    strcat(mesaj, isim);
16:
17:    printf("%s\n",mesaj);
18:
19:   return 0;
20: }
```

ÇIKTI

```
Adiniz ? : Mert
Selam Mert
```

Program 12.7: `strlen` *fonksiyonunun kullanımı*

```
01: /* 12prg07.c: Bir karakter dizisinin uzunluğunu bulur
02: */
03:
04: #include <stdio.h>
```

```
05: #include <string.h>
06:
07: int main()
08: {
09:    char s[20];
10:    int  k = 0;
11:
12:    printf("Bir seyler yazin : ");
13:    scanf("%s",s);
14:
15:    /* sonlandırıcı karaktere kadar */
16:    while( s[k] != '\0' )
17:          k++;
18:
19:    puts("Dizinin uzunlugu");
20:    printf("strlen kullanarak  = %d\n",strlen(s));
21:    printf("strlen kullanmadan = %d\n",k);
22:
23:  return 0;
    }
```

ÇIKTI

```
Bir seyler yazin : deneme stringi
Dizinin uzunlugu
strlen kullanarak  = 14
strlen kullanmadan = 14
```

Program 12.8: *Isim sırlama*

```
01: /* 12prg08.c
02:    Kabarcık Sıralama (Bubble Sort) Algoritması ile
03:    isimleri alfabetik sırayla listeler */
04:
05: #include <stdio.h>
06: #include <string.h>
07:
08: #define n  5
09:
10: int main()
11: {
12:   char isim[n][8] = { "Semra", "Mustafa", "Ceyhun",
13: "Asli", "Leyla" };
14:   char  gecici[8];
15:   int i,j,k;
16:
17:   puts("Once:\n------");
18:   for(i=0; i<n; i++)
19:      printf("%s\n",isim[i]);
20:
21:   /* sırala */
22:   for(k=0; k<n-1; k++)
23:   for(j=0; j<n-1; j++)
24:     if( strcmp(isim[j],isim[j+1]) > 0 )  /*
25: isim[j]>isim[j+1] ? */
```

```
26:       {
27:         strcpy(gecici   ,isim[j]);
28:         strcpy(isim[j]  ,isim[j+1]);
29:         strcpy(isim[j+1],gecici);
30:       }
31:
32:     puts("\nSonra:\n------");
33:     for(i=0; i<n; i++)
34:        printf("%s\n",isim[i]);
35:
      return 0;
    }
```

ÇIKTI

```
Once:
------
Semra
Mustafa
Ceyhun
Asli
Leyla

Sonra:
------
Asli
Ceyhun
Leyla
Mustafa
Semra
```

12.5 Katarların Fonksiyonlarda Kullanılması

Katarların fonksiyonlara parametre olarak geçirilmesi durumuna sıklıkla rastlanır. Gerçekte fonksiyona parametre olarak aktarılan karakter dizisini gösteren bir adrestir. Bu yüzden karakter dizileri fonksiyonlara çoğunlukla gösterici tipinde geçirilir.

Aşağıdaki iki örnekte yazılan `struzn` ve `strcev` fonksiyonları sırasıyla `strlen` ve `strrev` fonksiyonların dengi niteliğindedir. Burada kullanılan benzer mantıkla, `string.h` kütüphanesindki birçok fonksiyon yazılabilir. İnceleyiniz.

Program 12.9: `strlen` dengi bir fonksiyon: `struzn`

```
01: /* 12prg09.c: Bir katarın uzunluğunu bulan strlen
02: dengi bir fonksiyon */
03:
04: #include <stdio.h>
05: #include <string.h>
06:
07: int struzn(char *);
08:
09: int main()
10: {
11:   char *s;
12:
13:   printf("Bir katar girin: ");
```

```
14:    gets(s);
15:
16:    printf("Uzunlugu (struzn) : %d\n",struzn(s));
17:    printf("Uzunlugu (strlen) : %d\n",strlen(s));
18:
19:   return 0;
20:  }
21:
22:  /* bir karakter dizisinin uzunluğunu hesaplar */
23:  int struzn(char *str)
24:  {
25:    int n = 0;
26:
27:    while(str[n])
28:       n++;
29:
30:    return n;
     }
```

ÇIKTI

```
Bir katar girin: Programlama
Uzunlugu (struzn) : 11
Uzunlugu (strlen) : 11
```

Program 12.10: `strrev` dengi bir fonksiyon: `strcev`

```
01:  /* 12prg10.c: Bir katarın tersini veren bir fonksiyon
02:  */
03:
04:  #include <stdio.h>
05:  #include <string.h>
06:
07:  char *strcev(char *);
08:
09:  int main()
10:  {
11:    char s[50];
12:
13:      printf("Bir katar girin: ");
14:      scanf("%s",s);
15:
16:      printf("Katar, s : %s\n",s);
17:      printf("Tersi, strcev(s) : %s\n",strcev(s));
18:
19:   return 0;
20:  }
21:
22:  /* str katarını ters-yüz eder */
23:  char *strcev(char *str)
24:  {
25:    int  i,n;
26:    char gecici;
27:
28:    n = strlen(str);
```

```
29:
30:    for(i=0; i<n/2; i++)
31:    {
32:       gecici     = str[i];
33:       str[i]     = str[n-i-1];
34:       str[n-i-1] = gecici;
35:    }
36:
37:    return str; /* geri dönüş değeri bir gösterici */
    }
```

ÇIKTI

```
Bir katar girin: Programlama
Katar, s : Programlama
Tersi, strcev(s) : amalmargorP
```