

# Bölüm 9
# Dizgiler

- Dizgi Tanımı
- Dizgi Girdi İşlemleri
- Dizgi Çıktı İşlemleri
- Dizgi Fonksiyonları
- Karakter Fonksiyonları
- Gösterge Dizgileri

# Dizgi Tanımı

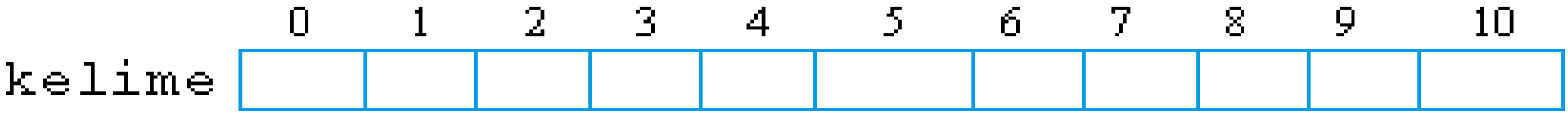
İki çift tırnak işareti “ ve ” içinde tanımlanmış olan sıralı karakterler bütününe ***dizgi*** (string) adı verilmektedir.

| **Dizgi** | **Açıklama** |
|---|---|
| “Merhaba” | 7 karakter içeren bir dizgi |
| “Bu bir dizgi” | 12 karakter içeren dizgi. |
| “B” | Bir karakter içeren bir dizgi |
| “” | Boş dizgi |

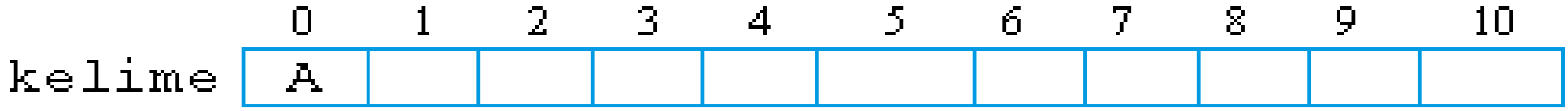
# Dizgi Tanımı

**char** *dizi_adı*[*uzunluk*] ;

```
char kelime[11];
```

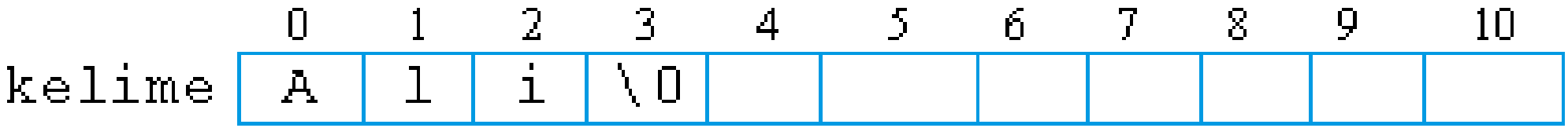



```
kelime[0] = 'A';
```



# Dizgi Tanımı

```
kelime[1] = 'l';
kelime[2] = 'i';
kelime[3] = '\0';
```



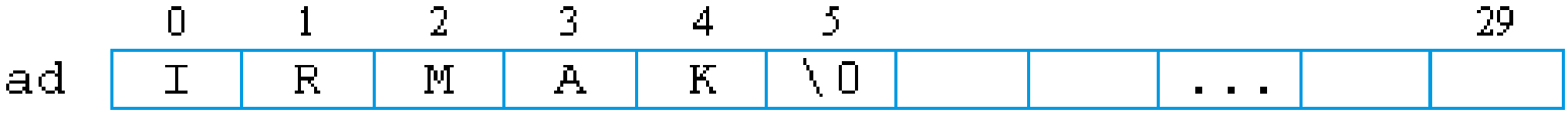

Bir dizginin sonu ***boş karakter*** (NULL character) olan `'\0'` karakteri ile biter.

# Dizgi Tanımı

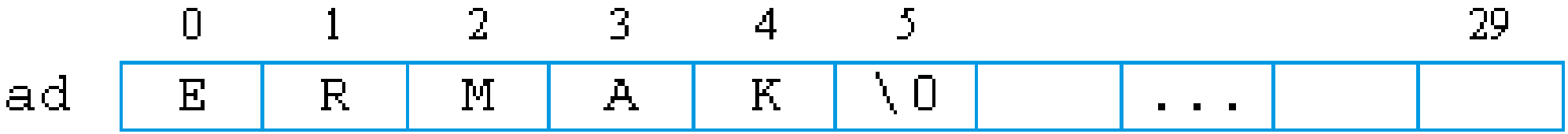
Dizgileri tanımlarken ilk değerini de atayabiliriz.

**`char` *dizi_adı*[*uzunluk*] = *dizgi_sabiti*;**

```
char ad[30]="IRMAK";
```




```
ad[0]='E';
```

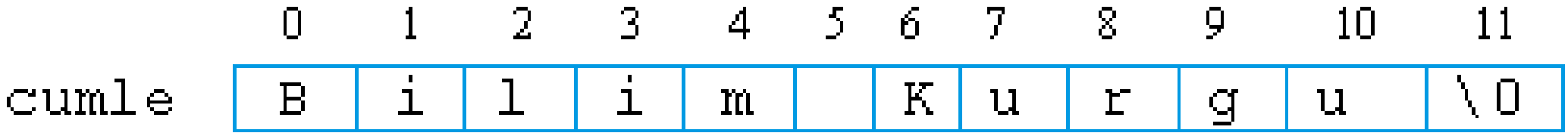


# Dizgi Tanımı

Dizgi tanımlamalarını, dizgi uzunluğunu dizi tanımlaması sırasında verilmeden ve ilk değerini atayarak da yapabiliriz

**`char`** ***dizgi*** **[] =** ***dizgi_sabiti***;

```
char cumle[] = "Bilim Kurgu";
```

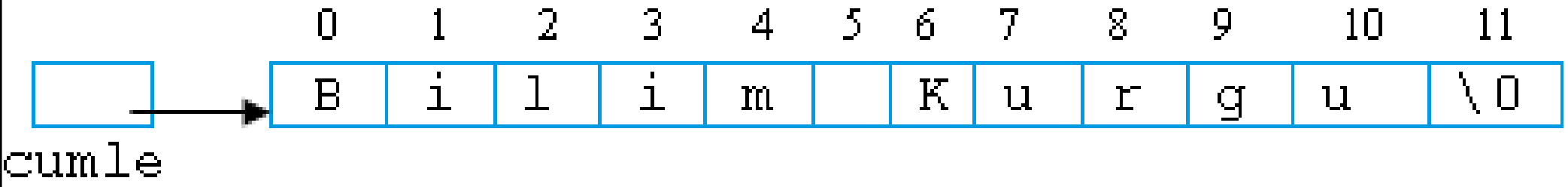


# Dizgi Tanımı

Dizgiler tanımlanırken göstergeler kullanılarak da aşağıdaki gibi tanımlanabilir. Çünkü her bir dizgi aslında bir dizi ile tanımlanmıştır.

**char ****dizgi_adı* **=** *dizgi_sabiti*;

```
char *cumle = "Bilim Kurgu";
```



# Dizgi Tanımı

**Örnek:**

```
char cumle[] = "Merhaba Dunya";
int say = 0;
int i;
for (i = 0; cumle[i] != '\0'; i++)
    say++;
printf("%s %d karakter icerir.",
            cumle, say);
```

**Çıktı:**

```
Merhaba Dunya 13 karakter icerir.
```

# Dizgi Girdi İşlemleri

**`scanf()`** fonksiyonu girilen değerler içinde boşluk veya enter işareti ( ↵) görünceye kadar okuma işine devam eder

```
scanf ("%s", dizgi_adı);
```

**Örnek:**

```
char kelime[11];
scanf ("%s", kelime);
```

**Girdi**

```
Bilgisayar↵
Bilge Sor.↵
Bil↵
```

**kelime**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| B | i | l | g | i | s | a | y | a | r | \0 |
| B | i | l | g | e | \0 | | | | | |
| B | i | l | \0 | | | | | | | |

# Dizgi Girdi İşlemleri

```
char kelime[11];
scanf ("%7s", kelime);
```

| Girdi | kelime | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Programlama↵ | P | r | o | g | r | a | m | \0 | | | |
| Prog.↵ | P | r | o | g | . | \0 | | | | | |

# Dizgi Girdi İşlemleri

**`gets()`** fonksiyonu enter ya da girdi sonunu belirleyen (ctrl+z) karakterini görünceye kadar girdiyi okumaya devam eder ve okuduğu değerin sonuna boş karakterini '\0' otomatik olarak ekleyerek *dizgi_adı*'na bu değerleri atar.

**`gets`** (***dizgi_adı***);

**Örnek:**

```
char cumle[15];
gets (cumle);
```

Girdi

␣␣iyi␣misin?↵

cumle

| ␣ | ␣ | i | y | i | ␣ | m | i | s | i | n | ? | \0 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

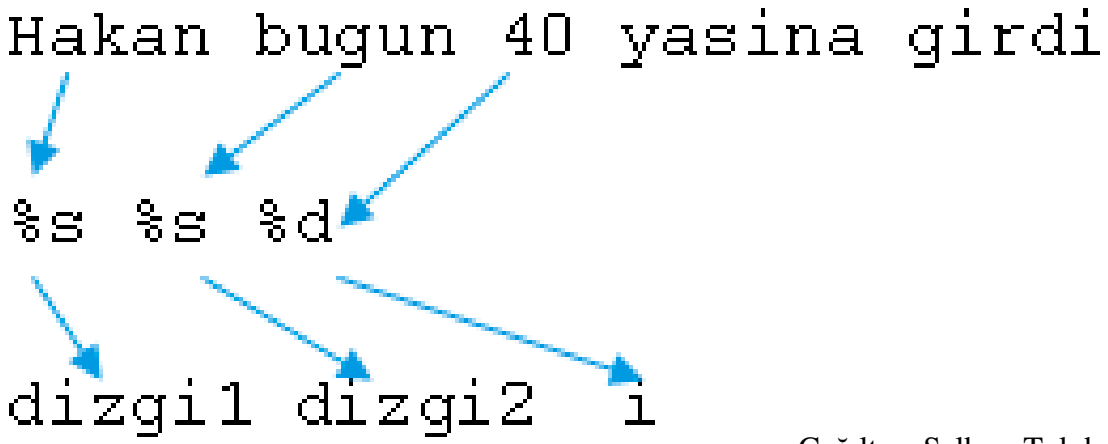
# Dizgi Girdi İşlemleri

**`sscanf()`** fonksiyonu kullanıldığında girdi bilgisi klavyeden değil bir başka dizgiden alınır.

**`sscanf`** (*dizgi_adı, format_dizgisi, girdi_listesi*);

**Örnek:**

```
char cumle[]="Hakan bugun 40 yasina girdi";
char dizgi1[20], dizgi2[20];
int i;
sscanf cumle,"%s %s %d",dizgi1,dizgi2,&i);
printf ("%s --> %d\n",dizgi1, i);
```



# Dizgi Çıktı İşlemleri

**printf()** fonksiyonunu dizgilerin bastırılması amacıyla da kullanabiliriz.

**printf** (“*%s*”, *dizgi_adı*);

**Örnek:**

```
char dizgi1[15]= “merhaba”;
char dizgi2[]= “iyi”;
```

| Komut | Çıktı |
|---|---|
| `printf (“%s”, dizgi1);` | merhaba |
| `printf (“%s”, “Nasilsin?”);` | Nasilsin? |
| `printf (“%5s”, dizgi2);` | □□iyi |
| `printf (“%-5s”, dizgi2);` | iyi□□ |
| `printf (“%s %s”, dizgi1, dizgi2);` | merhaba iyi |

# Dizgi Çıktı İşlemleri

**puts()** fonksiyonu standart çıktı birimine yani ekrana dizginin değerinin bastırılmasını sağlar ve daha sonra yeni satır karakterini otomatik olarak çıktının sonuna ekler.

**puts** (*dizgi_adı*);

**Örnek:**

```
char dizgi1[15]= “merhaba”;
char dizgi2[]= “iyi”;

char dizgi1[]="merhaba";
char dizgi2[]="nasilsin?";
puts(dizgi1);
puts(dizgi2);
```

**Çıktı:**

```
merhaba
nasilsin?
```

# Dizgi Çıktı İşlemleri

**`sprintf()`** fonksiyonu farklı değişkenlerin değerini belirli bir format dizgisine uygun olarak yeni bir dizginin içine kopyalar.

**`sprintf`** (*dizgi_adı, format_dizgisi, liste*);

**Örnek:**

```
float benzin = 47.0;
float km = 300;
char benzin_km[80];
sprintf(benzin_km,"km. de %5.3f lt.
        benzin yakiyor", km/benzin);
printf ("%s ", benzin_km);
```

**Çıktı:**

```
km. de 6.383 lt. benzin yakiyor
```

# Dizgi Fonksiyonları

Dizgi işleme amacıyla hazırlanmış programlarda kolaylık sağlayabilecek bir çok fonksiyon **<string.h>** kütüphanesi içinde tanımlanmıştır. Bir dizginin içindeki karakter sayısını bulmak için **strlen()** fonksiyonu kullanılır.

**strlen (*dizgi_adı*);**

**Örnek:**

```
int uzunluk;
char dizgi[10] = "Ali"
uzunluk = strlen(dizgi);
```

↓

```
3
```

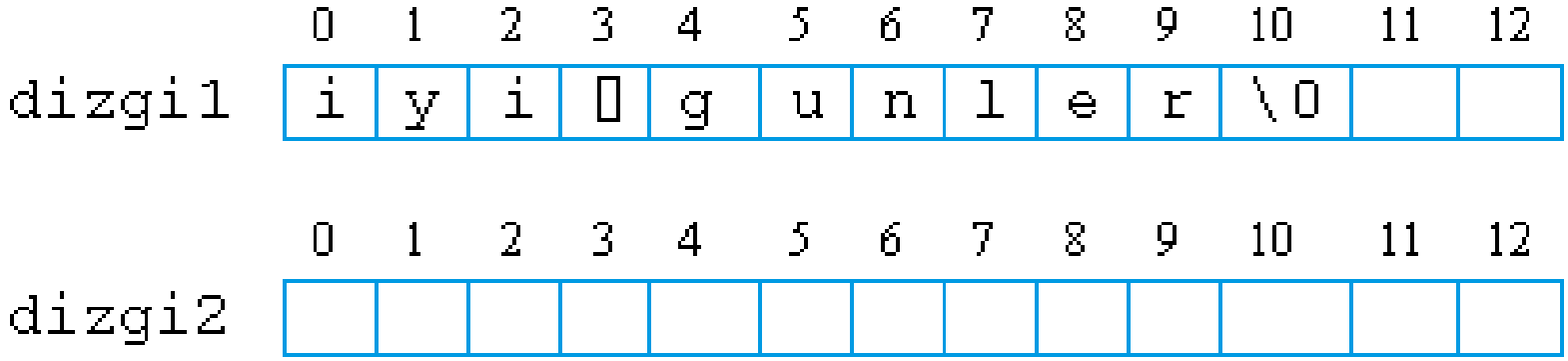
# Dizgi Fonksiyonları

**`strcpy()`** fonksiyonu dizgi kopyalama fonksiyonudur.

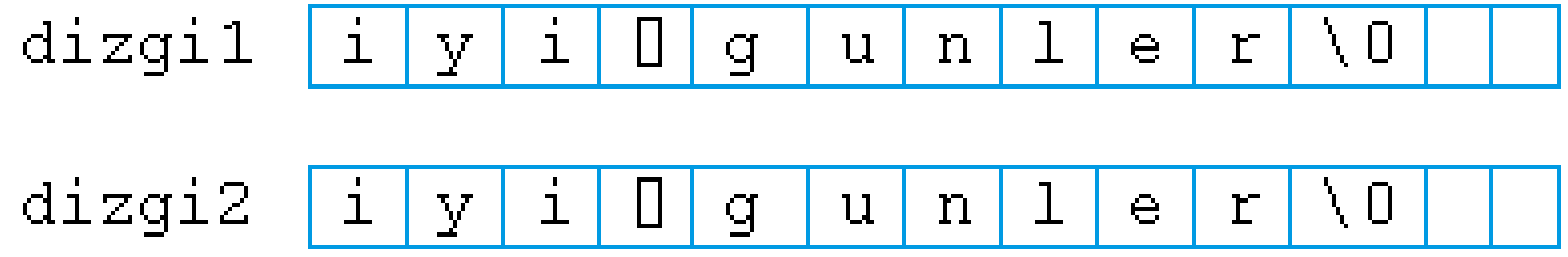
**`strcpy`** (***dizgi2_adı, dizgi1_adı***);

**Örnek:**

```
char dizgi1[13]="iyi gunler";
char dizgi2[13];
```




```
strcpy (dizgi2, dizgi1);
```



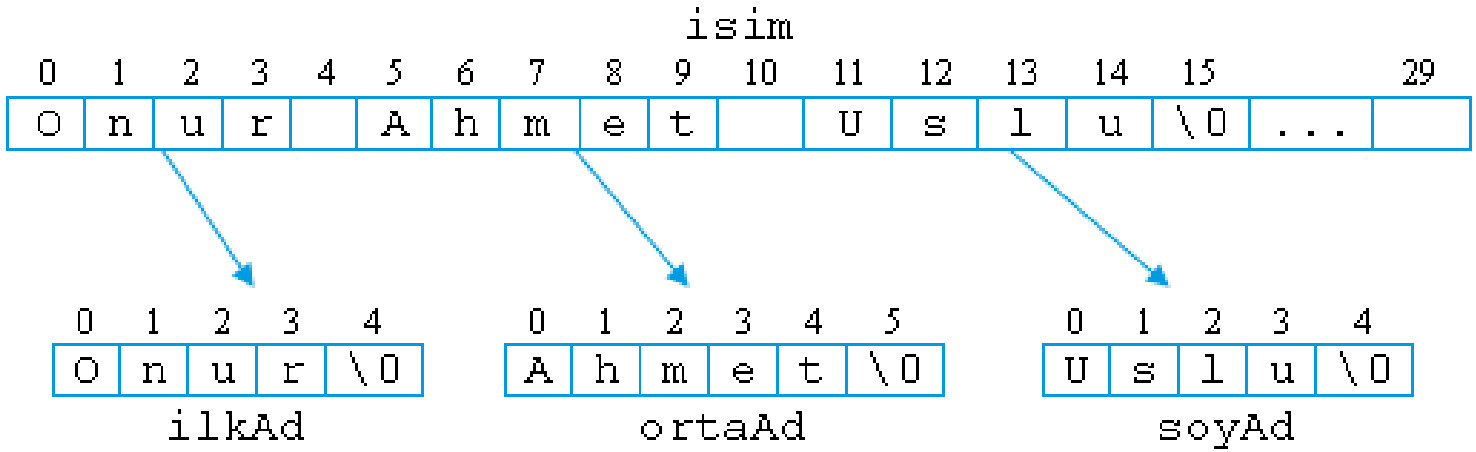
# Dizgi Fonksiyonları

**`strncpy()`** fonksiyonu *dizgi1_adı*'nın içindeki ilk *n* karakterin *dizgi2_adı*'na kopyalanmasını sağlar.

**`strncpy (`*dizgi2_adı, dizgi1_adı, n*`);`**

**Örnek:**

```
char isim[30]="Onur Ahmet Uslu";
char soyAd[10], ilkAd[10], ortaAd[10];
```




```
strncpy(ilkAd, isim,4); ilkAd[4]='\0';
strncpy(ortaAd, &isim[5], 5);
ortaAd[4]='\0'; strcpy(soyAd, &isim[11]);
```

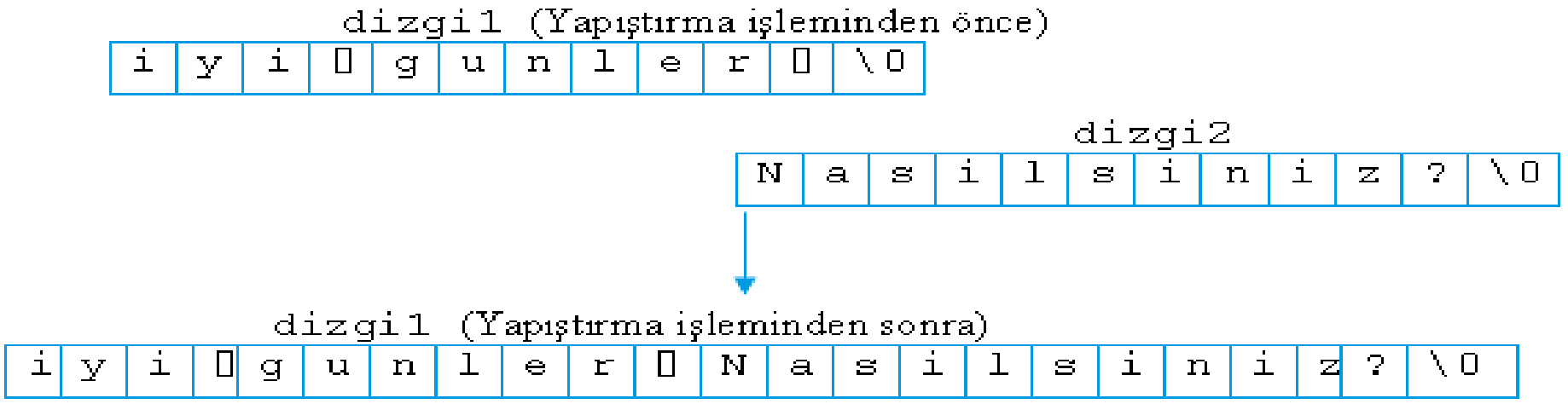
# Dizgi Fonksiyonları

**`strcat()`** fonksiyonu bir dizginin sonuna diğer bir dizginin yapıştırılmasını sağlar.

**`strcat (`*dizgi1_adı, dizgi2_adı*`);`**

**Örnek:**

```
char dizgi1[12]="iyi gunler ";
char dizgi2[12]= "Nasilsiniz?";
strcat (dizgi1, dizgi2);
printf ("\ndizgi 1: %s %d",dizgi1, strlen(dizgi1));
printf ("\ndizgi 2: %s ",dizgi2);
```



# Dizgi Fonksiyonları

**strncat()** fonksiyonu *dizgi2_adı*'nın ilk *n* karakterinin *dizgi1_adı*'nın sonuna yapıştırılmasını sağlar.

**strncat (*dizgi1_adı, dizgi2_adı, n*);**

**Örnek:**

```
char dizgi1[15]="iyi gunler ";
char dizgi2[15]= "Nasilsiniz?";
strncat (dizgi1, dizgi2, 5);
printf ("\ndizgi 1: %s ",dizgi1);
printf ("\ndizgi 2: %s ",dizgi2);
```

**Çıktı:**

```
dizgi 1: iyi gunler Nasil
dizgi 2: Nasilsiniz?
```

# Dizgi Fonksiyonları

`strcmp()` fonksiyonu iki dizginin karşılaştırılmasını sağlar.

**`strcmp`** (***dizgi1_adı, dizgi2_adı***);

| dizgi1 | dizgi2 | strcmp(dizgi1, dizgi2) |
|---|---|---|
| Balik | Kapi | Negatif |
| Kapi | Balik | Pozitif |
| Kapi | Kapi | 0 |
| Kapi | Kapici | Negatif |
| Kapi | kapi | Pozitif |

# Dizgi Fonksiyonları

**`strncmp()`** fonksiyonu iki dizginin ilk n karakterlerinin karşılaştırılmasını sağlar.

**`strncmp`** (*dizgi1_adı, dizgi2_adı, n*);

**Örnek:**

```
char dizgi1[13]="iyi gunler ";
char dizgi2[13]= "iyi misiniz?";
printf ("\n%d ",strncmp (dizgi1, dizgi2, 3));
```

**Çıktı:**

```
0
```

# Dizgi Fonksiyonları

**strstr()** fonksiyonu bir dizginin içinde diğer bir dizgiyi arar.

**strstr(*dizgi1_adı, dizgi2_adı*);**

**Örnek:**

```
char dizgi1[13]="iyi gunler ";
char dizgi2[13]= "gun";
if (strstr (dizgi1, dizgi2) == '\0')
   printf ("dizgi2 dizgi1 in icinde YOK");
else
   printf ("dizgi2 dizgi1 in icinde VAR");
```

**Çıktı:**

```
dizgi2 dizgi1 in icinde VAR
```

# Dizgi Fonksiyonları

**Örnek:** Kullanıcının girdiği bir dizgiyi okuyarak, bu dizginin tersini bulan bir program yazınız.

```
#include <string.h>
 int main(void)
 {
   char str2[30], str1[30];
   int i, uzunluk;
   printf("Bir dizgi giriniz:");
   gets(str1);
   uzunluk=strlen(str1);
   for(i=0;i<=uzunluk;++i)
     strncpy(&str2[i],&str1[uzunluk-i-1],1);
   printf("%s",  str2);
   return(0);
 }
```

**Çıktı:**

```
Bir dizgi giriniz:kitap
patik
```

# Karakter Fonksiyonları

```
char k;
```

| Fonksiyon | Döndürdüğü Değer |
|---|---|
| `isalpha(k)` | k bir harf ise, sıfırdan farklı, diğer durumlarda sıfır |
| `isdigit(k)` | k bir sayı ise, sıfırdan farklı, diğer durumlarda sıfır |
| `islower(k)` | k küçük bir harf ise, sıfırdan farklı, diğer durumlarda sıfır. |
| `isupper(k)` | k büyük bir harf ise, sıfırdan farklı, diğer durumlarda sıfır |
| `isspace(k)` | k boşluk, yeni satır veya tab karakteri gibi bir karakterse, sıfırdan farklı, diğer durumlarda sıfır |
| `tolower(k)` | k'nın değerinin küçük harf karşılığı |
| `toupper(k)` | k'nın değerinin büyük harf karşılığı |

# Karakter Fonksiyonları

**Örnek:** Kullanıcıdan isimler ve notların karışık olarak girildiği bir dizgi alan ve bu dizginin içindeki isimleri ekranda gösteren bir program yazınız.

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
int main(void)
{  char str1[70];
   int k, i;
   printf("Bir dizgi giriniz:");
   gets(str1);
   k=strlen(str1);
   for(i=0;i<=k;++i)
     if(isalpha(str1[i]))
       printf("%c", str1[i]);
   return(0);
}
```

**Çıktı:**

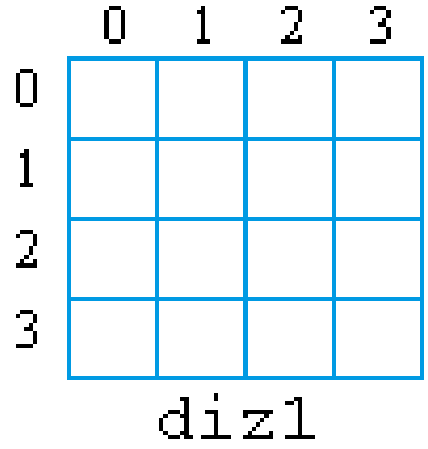
```
Bir dizgi giriniz:O12s34m6a7n 100
Osman
```

# Gösterge Dizgileri

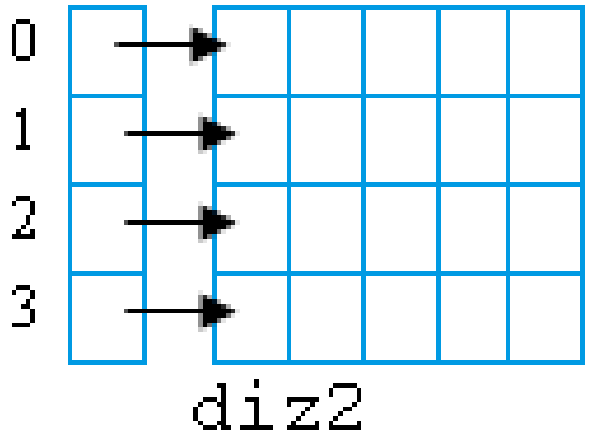
Diziler ve göstergeleri kullanarak da dizgileri tanımlayabiliriz.

**Örnek:**

```
char diz1[4][4];
```




```
char *diz2[4];
```



# Gösterge Dizgileri

**Örnek:**

```
char mevsimler[4]={"Sonbahar",
                   "Kis",
                   "Ilkbahar",
                   "Yaz"};
```

