STRING (Harf dizinleri)

Harf dizinleri metin adıyla bildiğimiz türdür. C dilinde string'ler asıl veri türlerinden birisi değildir. Char türünden bir boyutlu arrayler olarak elde edilir. printf() ve scanf() fonksiyonlarının argümanları olarak çok kullanılırlar. C dilinin temel veri türlerinden olmadığı için, string'lerin kullanılış biçimi array'lerin kullanılışından farklı değildir; yani bir string'in öğelerine ancak tek tek erişmek olanağı vardır. Başka bir deyişle, C dili bir string'in bütün öğelerini bir bütün olarak isleyen operatörlere sahip değildir. Böyle olmakla birlikte, hemen her C derleyicisi yanında satılan fonksiyon kütüphanesinde string'lerle yapılabilecek bütün işleri yapmaya yarayan hazır fonksiyonlar vardır. String'lerin karşılaştırılması, kopyalanması, birinin ötekine eklenmesi v.b. işlemleri yapmak için çok sayıda C fonksiyonu vardır. Bunları bu kesimin sonunda inceliyeceğiz.

C dilinde string'ler aşağıdaki yöntemlerle kurulabilir:

1. Yöntem

İstenen metin " " simgeleri arasına yazılır. Örneğin,

```
printf("C Programlama dili çok hünerlidir.");
```

deyiminde " " işaretleri arasına alınan metin bir harf dizini (metin,string) oluşturur. Buradaki dizin printf() fonksiyonunun argümanı (bağımsız değişkeni) rolünü oynar. " " içindeki bir metnin son harfinden sonra, C derleyicisi metnin sonuna gelindiğini belirtmek üzere \0 karakterini koyar. Standart cikista (ekran, printer) görunmeyen bu karaktere NULL terminatör adı verilir ve enter tuşuna basıldığında derleyici tarafından kendiliğinden oluşturulur.

2. Yöntem

İstenen metni içerebilecek bir yeri ana bellekte ayırmak için, yeterli büyüklükte bir array yaratılır. Örneğin,

```
char metin[20] ;
```

bildirimi 20 öğeli bir array yaratır. Herbir öğe bir harf temsil edecek bir değişkendir. Bu değişkene değer adamak için

```
metin[] = "Merhaba C." ;
```

yazmak yeterlidir. Bu adama, gerçekte

```
metin[0] = 'M' ;

metin[1] = 'e' ;
```

```
metin[2] = 'r' ;

metin[3] = 'h' ;

metin[4] = 'a' ;

metin[5] = 'b' ;

metin[6] = 'a' ;

metin[7] = ' ' ;

metin[8] = 'C' ;

metin[9] = '.' ;

metin[10] = '\0';
```

adamalarının yapılmasına denktir. Burada '\0' simgesi NULL (bos) karakter adıyla anılan değerdir. String'in bittiğini belirtir.

Bu işlemi daha kısa olarak

```
char metin[] = "Merhaba C.\n" ;
```

biçiminde de yazabiliriz. Bu halde, metin adlı array'e bildirim sırasında değer adanmaktadır. Aynı iş, daha zor biçimde

```
char metin[]={'M','e','r','h','a','b','a',' ','C''.','\0'};
```

biçimde de yapılabilir.

3. Yöntem

String bildiriminde pointerlerden yararlanabiliriz. Örneğin,

```
char *soyad ;
```

bildiriminden sonra

```
soyad = "Deniz";
```

adaması geçerlidir. Bu adamada 'D' harfi soyad pointerinin gösterdiği adrese yerleşir. Öteki harfler , o adresten sonraki 1 er byt'lık bellek hücrelerine sırayla yerleşirler. Dolayısıyla,

```
*soyad == 'D'

*(soyad +1) == 'e'

*(soyad +2) == 'n'
```

```
*(soyad +3) == 'i'

*(soyad +4) == 'z'

*(soyad +5) == '\0'
```

deyimleri doğrudur.

Pointerlerin ve arrayler'in kullanılışını gösterebilmek için, bir metni kopya etmeye yarayan strcpy() fonksiyonunu hem array yardımıyla hem pointer yardımıyla olmak üzere iki türlü tanımlıyacağız. Aynı işi gören bu iki fonksiyonun ikinci türünün daha kolay tanımlandığını göreceksiniz:

1. Array Yardımıyla Tanımlama

```
strcpy(s,t)

char s[], t[];

{

int i;

i=0;

while((s[i] = t[i]) != '\0')

i++;

}
```

2. Pointer Yardımıyla Tanımlama I

```
strcpy(s,t)

char *s,*t;

{

while((*s = *t) != '\0' )

{

s++;

t++;

}

}
```

Artırma operatörünü kullanarak, bu fonksiyonu daha kısa olarak da yazabiliriz:

3. Pointer Yardımıyla Tanımlama II

```
strcpy(s,t)

char *s,*t;

{

while((*s++ = *t++) != '\0' )

;

}
```

Pointerler arasında çıkarma işlemi de yapılabilir. Örneğin, p ve q aynı bir array'in farklı iki öğesini işaret ediyorlarsa, p - q deyimi array'in p-inci öğesi ile q-inci öğesi arasında kaç öğe olduğunu belirtir. Derleyici, array'in iki öğesinin işaret ettiği adreslerin numaraları arasındaki farkı bulur ve bu farkin pointerin işaret ettiği öğe türüne göre, kaç array öğesine karşılık geleceğini hesaplar. Örneğin, bir metnin (string) uzunluğunu bulan C kütüphane fonksiyonunu düşünelim. Bu fonksiyonun tanımı aşağıdaki gibidir:

```
strlen(s)

char *s;

{

char *p = s;

while (*p != '\0')

p++;

return(p-s);

}
```

Burada p pointerinin ilk değeri s olarak verilmiştir. Dolayısıyla, p pointeri metnin ilk harfinin adresini işaret etmektedir. Ondan sonra, metin sonunu belirleyen NULL '\0' karakteriyle buluşana dek, while döngüsü p pointerini 1 er artırmaya devam edecektir. p pointeri char türündendir. char için ayrılan bellek bölgesi 1 byte uzunlukta olduğuna göre, p den p++ değerine geçiş bellek numarasını 1 artıracaktır. Böylece p-s sayısı metnin kaç byte (yani kaç harf) uzunlukta olduğunu belirtmiş olacaktır.

C dili string'leri doğrudan doğruya işlemez. İstenilen işlemi yapacak fonksiyonların yazılması gerekir. Birçok C derleyicisinin yanında bu işleri yapacak standart

fonksiyonlar satılır. Bunları programcı kendisi de yaratabilir. En çok kullanılan bazı string fonksiyonları aşağıda incelenmiştir :

strcat()

Bir string'i ötekinin arkasına ekler.

```
char *strcat(char *dizin1, char *dizin2);
```

Bu fonksiyon yürütüldüğünde, dizin2 string'i dizin1 string'inin arkasına eklenir.

strcmp()

Verilen iki string'i karşılaştırır.

```
int strcmp(char *dizin1, char *dizin2);
```

Bu fonksiyon dizin1 ile dizin 2 yi karşılaştırır. Eğer ikisi eşitse strcmp() fonksiyonu 0 değerini alır; dizin1, dizin2 den büyükse pozitif bir değer; küçükse negatif bir değer alır. Karşılaştırma yöntemi şöyledir: Harf harf karşılaştırdığı iki dizinde, farklı harflerle karşılaştığında dizin2 deki harfin ASCII kodunu dizin1 deki harfin ASCII kodundan çıkarır.

strcmp() fonksiyonunu array ya da pointer kullanarak tanımlamak olanağı vardır:

1. Array İle Tanımlama

```
strcmp(s,t)

char s[], t[];

{

int i;

i = 0;

while (s[i] == t[i])

if (s[i++] == '\n')

return(0);

return(s[i] - t[i]);

}
```

2. Pointer İle Tanımlama

```
strcmp(s,t)

char *s,*t;

{

for( ; *s == *t; s++, t++ )

if(*s == '\0' )

return(0);

return(*s - *t);

}
```

strcpy()

Bir string'i kopya eder.

```
char *strcpy(char *d1, char *d2);
```

Bu fonksiyon d2 stringini d1' e kopya eder. C derleyicisi d1'in d2 dizinini içerebilecek uzunlukta olup olmadığını denetlemez. Dolayısıyla, programcı bunu kendisi güvenceye almalıdır.

strlen()

Bir string'in uzunluğunu verir.

```
unsigned strlen(char *dizin);
```

Bu fonkiyon dizin içindeki harfleri sayar. Tanımı aşağıdaki gibidir:

```
strlen(s)

char *s;

{

char *p = s;

while ( *p != '\0' )

p++;

return(p - s);

}
```

strlwr()

Bir dizindeki büyük harfleri küçük harflere çevirir.

```
char *strlwr(char *dizin);
```

Bu fonksiyon dizin'in harflerini küçük harf olarak yazar.

strupr()

Bir dizindeki harfleri büyük harfe çevirir.

```
char *strupr(char *dizin);
```

KARAKTER DENETLEME FONKSİYONLARI

Aşağıdaki fonksiyonlar birçok C derleyicisinde fonksiyon kütüphanesinde olmak yerine macro olarak derleyicide yer almıştır. Bazılarında <ctype.h> içinde yer almıştır.

Aşağıdaki fonksiyonlarda ch char türünde bir değişkeni gösteriyor varsayılacaktır.

isalnum()

```
isalnum(int ch);
```

Eğer ch bir harf yada sayı ise fonksiyon değeri sıfırdan farklı; değilse 0 olur.

isalpha()

```
int isalpha(int ch);
```

Eğer ch bir harf ise fonksiyon sıfırdan farklı bir değer verir; değilse 0 verir.

isdigit()

```
int isdigit(int ch);
```

islower()

```
int islower(int ch);
```

ch harfi küçükse sıfırdan farklı; değilse 0 değeri verir.

isupper()

```
int isupper(int ch);
```

ch harfi büyükse sıfırdan farklı; değilse 0 değeri verir.

isascii()

```
int isascii(int ch);
```

Eğer ch nin ASCII kodu 0-127 arasında ise fonksiyon değeri sıfırdan farklı; değilse 0 dir.

BELLEK AYIRAN FONKSİYONLAR

sizeof()

```
int sizeof(x);
```

x yerine herhangi bir değişken ya da asıl veri türlerinden birisi konulduğunda, donanım sisteminin o değişkene ya da veri türüne ayırdığı bellek büyüklüğünü verir.

malloc()

```
char *x;

x = malloc(n); /* n istenen byte sayısıdır */
```

Operatör sisteminin istenen bir bellek alanını bir pointer'e tahsis etmesini sağlar. Örneğin, donanım sisteminin float veri türü için ayırdığı bellek alanının 1000 katı bir yeri yapı adlı pointere tahsis etmek için aşağıdaki program parçası kullanılabilir:

```
char *malloc();

char *yapi;

int gerekli;

gerekli = sizeof(float)*1000;

yapi = malloc(gerekli);
```

free()

malloc() fonksiyonu ile bir değişkene tahsisi edilen bellek alanını serbest kılar.

```
int free(pointer);
```

Burada pointer değişkenine daha önce bellek alanı tahsisi edilmiş olmalır.

Örneğin,

```
char *bellek, *malloc();

bellek = malloc; /* bellek tahsisi yapılıyor */

...

free(bellek); /* tahsis edilmiş bellek serbest bırakılıyoryor
*/
```