# DİZGİLER

## Dizgi tanımlama

C dilindeki dizgi, dizideki son karakteri '`\0`' yani NULL karakteri (NULL character) olan, `char` veri tipinde tek boyutlu bir dizidir. Bu özel dizilerde metinler tutulur. En yaygın dizgi tanımı aşağıdaki şekilde yapılır;

```
char metin[] = "CLang";
```

Bu tanımda çift tırnak (") karakterleri kullanılarak ilk değer olarak metin kimlikli dizgiye "Clang" metin değişmezi (string literal) atanmıştır. Aslında karakter dizisi olan bu metin aşağıdaki iki şekilde de tanımlanabilir;

```
char metin1[]={'C','L','a','n','g','\0'}; //yada
char metin2[]={'C','L','a','n','g', 0 };
```

Dizgiler karakter göstericisi olarak da tanımlanır. Bu durumda gösterici dizinin ilk elemanını gösterir;

```
char* metin3="CLang";
```

## Konsola Metin Yazma

Dizgiler, karakter dizileri olduğunda döngü kullanarak ekrana aşağıdaki şekilde yazabiliriz;

```
for (int i = 0; i < 5; i++) {
    printf("%c", metin[i]);
}
//yada
char* ptrc=&metin;
while (ptrc++!=0)
    printf("%c",*ptrc);
```

Döngü kullanmadan konsola metin yazmak için `printf` fonksiyonunda `%s` yer tutucusu kullanılır.

```
#include <stdio.h>
int main() {
    char metin[]="Clang"; //Metin değişmezler (string literal) çift tırnak arasında verilir.
    char metin1[]={'C','L','a','n','g','\0'}; //ya da
    char metin2[]={'C','L','a','n','g', 0 };
    char* metin3="CLang";

    //for döngüsü ile konsola metin yazma
    for (int i = 0; i < 5; i++)
        printf("%c", metin[i]);
    printf("\n");

    //while döngüsü ile konsola metin yazma
    char* ptrc=&metin[0]; //char *ptr3=metin3;
    while (*ptrc!=0)
        printf("%c",*ptrc++);
    printf("\n");

    //%s yer tutucusu ile konsola metin yazma
    printf("%s\n", metin);
    return 0;
}
```

## Klavyeden Metin Okuma

Klavyeden metin okunacak ise ilk önce okunacak karakter sayısına bağlı olarak bir dizgi tanımlanmalıdır;

```
char string[20];
```

Dizgileri klavyeden okumak için `scanf` fonksiyonlarında da `%s` yer tutucusu kullanılır. Belirlenen karakter sayısından fazlası okunduğunda belleğin istenmeyen bölgelerine ulaşılacağından okunacak metnin koyulacağı dizginin boyutuna dikkat edilmelidir.

`scanf` fonksiyonuna okunan değerlerin konulacağı değişkenin adresi `&` karakteri ile verilir. Ancak dizgi olan karakter dizileri `char[]` ile karakter göstericileri `char*` derleyici için aynı olarak kabul edilir. Bu nedenle okunacak dizgi değişkeninin önünde `&` karakteri kullanılmaz. Bu fonksiyonda dizgileri okumak için `%s` yer tutucusu kullanılır.

```
#include <stdio.h>
#include <string.h>
int main (){
   char adi[20];
   char soyadi[20];
   printf("Adınızı Giriniz:");
   scanf("%s", adi);
   printf("Soyadınızı Giriniz:");
   scanf("%s", soyadi);
   printf("Girdiğiniz değerler:");
   printf("%s-%s\n", adi,soyadi);
   return 0;
}
```

# Çok Kullanılan Dizgi Fonksiyonları

C dili, dizgiler üzerinde çeşitli işlemler gerçekleştirmek için standart `string.h` başlık dosyasına sahiptir. Bu kütüphanedeki fonksiyonları kullanırken göstericilerin bir karakter dizinini işaret etmesi gerektiği unutulmamalıdır.

| Fonksiyon | Açıklama |
|---|---|
| ... | |
| strcat() | Bir dizgiyi diğerinin sonuna ekler |
| strchr() | Dizgi içinde aranan bir karakterin ilk bulunanına ait göstericiyi döndürür. |
| strcmp() | ASCII olarak iki dizgiyi karşılaştırır. |
| strcpy() | Hafızada yer alan iki dizgiden birinin içeriğini diğerine kopyalar. |
| strlen() | Dizginin karakter uzunluğunu verir. |
| strrchr() | Dizgi içinde aranan bir karakterin son bulunanına ait göstericiyi döndürür. |
| strstr() | Dizgi içinde aranan ikinci bir dizginin ilk bulunanına ait göstericiyi döndürür. |
| strtok() | Dizgiyi verilen ayraçlara bağlı olarak alt dizgilere böler |
| ... | |

*Tablo 24. Çok Kullanılan Dizgi Fonksiyonları*

## strlen() fonksiyonu

Bu fonksiyon dizgideki karakter sayısını verir.

```
#include <stdio.h>
#include <string.h>
int main(){
   char *metin = "Merhaba";
   printf("Dizgi: %s\n", metin);
   printf("Dizgi Uzunluğu: %ld", strlen(metin));
   return 0;
}
```

## strcpy() fonksiyonu

Bu fonksiyon, ikinci parametredeki dizgiyi, birinci parametreye kopyalar. Burada dikkat edilmesi gereken husus; hedef gösterilen dizginin karakter sayısının veya göstericinin işaret ettiği bellek bölgesinin büyüklüğünün en az kaynak kadar olması gerektiğidir.

```
#include <stdio.h>
#include <string.h>
int main(){
   char* ptrToDegimmezDizgi = "Ne Haber?";
   char bosDizgi[20]; //Üzerine kopyalanacak metnin karakter sayısı 20 kabul edilmiş!
   char* ptr;
   strcpy(bosDizgi, ptrToDegimmezDizgi);
   printf("%s\n", bosDizgi);
   //strcpy(ptr, ptrToDegimmezDizgi);
   /* HATA: ptr sadece adres tutan değişkendir.
   Şu an için hiçbir dizgiyi göstermiyor! */
   ptr = bosDizgi;
   strcpy(ptr, ptrToDegimmezDizgi);
   printf("%s", ptr);
   return 0;
}
```

Bu fonksiyonu kullanmak yerine kendimiz de dizgi kopyalayan fonksiyon yazabiliriz;

```
#include <stdio.h>
void copyString(char*, char*);
int main(){
    char str1[100] , str2[100]="Deneme Metni";
    printf("str1: %s \nstr2: %s\n", str1,str2);
    copyString(str1, str1);
    printf("str1: %s \nstr2: %s\n", str1,str2);
    return 0;
}
void copyString(char* hedef, char* kaynak){
    int i = 0;
    for(i = 0; kaynak[i]!='\0'; i++)
        hedef[i] = kaynak[i];
    hedef[i] = '\0';
}
```

## strcat() fonksiyonu

Bu fonksiyon, parametre olarak girilen iki dizgiyi, birinci parametreye birleştirir.

```
#include <stdio.h>
#include <string.h>
int main() {
    char adi[] = "Ilhan";
    char* soyadi = "OZKAN";
    char adiSoyadi[50]; //Birleştirme sonrası oluşacak metnin karakter sayısı 50 kabul
edilmiş!
    strcat(adiSoyadi, adi);
    strcat(adiSoyadi, " ");
    strcat(adiSoyadi, soyadi);
    printf("%s", adiSoyadi); // Çıktı: Ilhan OZKAN
    return 0;
}
```

## strchr() fonksiyonu

Bu fonksiyon, birinci parametre olarak verilen dizgide, ikinci parametrede verilen karakteri arar. Bulunamaz ise bu fonksiyon NULL gösterici döndürür.

```
#include <stdio.h>
#include <string.h>

int main() {
  char dizgi[] = "Merhaba!";
  char aranan='h';
  char* ptr = strchr(dizgi, aranan);
  if (ptr != NULL) {
    printf("'%c' karakteri \"%s\" dizgisinde %ld indistedir.\n",
           *ptr, dizgi, ptr - dizgi);
  } else {
    printf("'%c' karakteri \"%s\" dizgisinde bulunamadı.\n",
           aranan, dizgi);
  }
  return 0;
}
```

## strcmp() fonksiyonu

Bu fonksiyon, parametre olarak girilen iki dizginin, karakterlerini karşılıklı olarak karşılaştırır. Karşılaştırılan iki karakter birbirinden farklı ise ASCII kodu farklarını döndürür. Tüm karakterler aynı ise sıfır döndürür.

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[] = "ARMUT";
    char str2[] = "armut";
    int result = strcmp(str1, str2);

    if (result == 0) {
        puts("İki metin de eşit.");
    } else if (result < 0) {
        puts("Birinci metin ikincisinden küçük");
    } else {
        puts("İkinci metin birincisinden küçük");
    }
    return 0;
}
// Çıktı: Birinci metin ikincisinden küçük
```

Alternatif olarak karşılaştırma fonksiyonunu aşağıdaki gibi yazabiliriz;

```
#include <stdio.h>
int compareString(const char*,const char*);
int main() {
    char str1[] = "Elma";
    char str2[] = "Elma2";
    int result = compareString(str1, str2);
    if (result == 0)
        printf("İki Metin Aynıdır.\n");
    else if (result < 0)
        printf("Birinci Metin İkincisinden Küçüktür:%d\n",result);
    else
        printf("Birinci Metin İkincisinden Büyüktür:%d\n",result);
    return 0;
}
int compareString(const char* metin1,const char* metin2) {
    while (*metin1 == *metin2) {
        if (*metin1 == '\0')  return (0);
        metin1++;
        metin2++;
    }
```

```
    return (*metin1 - *metin2);
}
// Birinci Metin İkincisinden Küçüktür:-50
```

# Karakter Fonksiyonları

C dilinde, standart olan bir başka başlık dosyası olan `ctype.h` içinde karakter fonksiyonları bulunur. Bu başlıkta dizgileri oluşturan karakterler üzerinde işlem yapmak için kullanılır. Karakterler aşağıdaki şekilde gruplandırılmıştır;

- Alfasayısal (alfanumeric): hem rakam hem harf barındıran karakter kümesidir.
- Beyaz boşluk: boşluk ‘ ’, yeni satır ‘`\n`’, satır başı ‘`\r`’, alt satır ‘`\v`’, ve tab ‘`\t`’ karakterlerinin oluşturduğu kümedir.
- Onaltılık Sayı Rakamları (hexadecimal digit): `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `a`, `b`, `c`, `d`, `e`, `f`

Aşağıda bu başlık dosyasında yer alan birçok fonksiyon yer almaktadır.

| Fonksiyon | Açıklama |
|---|---|
| `int isalnum(int c)` | Verilen karakter kodunun alfa sayısal olup olmadığını kontrol eder. |
| `int isalpha(int c)` | Verilen karakter kodunun harf olup olmadığını kontrol eder. |
| `int iscntrl(int c)` | Verilen karakter kodunun kontrol karakteri olup olmadığını kontrol eder. |
| `int isdigit(int c)` | Verilen karakter kodunun rakam olup olmadığını kontrol eder. |
| `int isgraph(int c)` | Verilen karakter kodunun konsola yazdırılabilir karakteri olup olmadığını kontrol eder. |
| `int islower(int c)` | Verilen karakter kodunun küçük harf olup olmadığını kontrol eder. |
| `int isprint(int c)` | Verilen karakter kodunun boşluk dahil konsola yazılabilir olup olmadığını kontrol eder. |
| `int ispunct(int c)` | Verilen karakter kodunun noktalama işareti olan karakterler olup olmadığını kontrol eder. |
| `int isspace(int c)` | Verilen karakter kodunun beyaz boşluk olup olmadığını kontrol eder. |
| `int isupper(int c)` | Verilen karakter kodunun büyük harf olup olmadığını kontrol eder. |
| `int isxdigit(int c)` | Verilen karakter kodunun onaltılık sayı rakamı olup olmadığını kontrol eder. |
| `int isblank(int c)` | Verilen karakter kodunun boşluk karakteri olup olmadığını kontrol eder. |
| `int toupper(int c)` | Verilen karakterin büyük harfinin kodunu döner. |
| `int tolower(int c)` | Verilen karakterin küçük harfinin kodunu döner. |

*Tablo 25. Ctype.h Karakter Fonksiyonları*

Bu fonksiyonlar kullanılarak küçük-büyük harf ayrımı yapmadan metinleri karşılaştıran bir program örneği verilmiştir;

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int compareString(char*, char*);
int main() {
    char str1[] = "Elma";
    char str2[] = "elma";
    int result = compareString(str1, str2);
    if (result == 0)
        printf("İki Metin Aynıdır.\n");
    else if (result < 0)
        printf("Birinci Metin İkincisinden Küçüktür.\n");
    else
        printf("İkinci Metin Birincisinden Küçüktür.\n");
    return 0;
}
int compareString(char* hedef, char* kaynak){
    int i = 0;
    for (i = 0; hedef[i]; i++) //hedefi küçük harfe çevir.
        hedef[i] = tolower(hedef[i]);
```

```
    for (i = 0; kaynak[i]; i++) //kaynağı küçük harfe çevir.
        kaynak[i] = tolower(kaynak[i]);
    return strcmp(hedef, kaynak);
}
```

# Metinden İlkel Veri Tiplerine Dönüşüm

Standart başlık dosyalarından olan `stdlib.h` başlık dosyasında daha önce işlediğimiz rastgele fonksiyonlarının yanında metinden ilkel veri tiplerine dönüşüm sağlayan bazı fonksiyonlar da bulunur.

| Fonksiyon | Açıklama |
|---|---|
| `double atof(const char*)` | Dizgi olarak verilen metni `double` tipine dönüştürür ve geri döndürür. |
| `int atoi(const char*)` | Dizgi olarak verilen metni `int` tipine dönüştürür ve geri döndürür. |
| `long atol(const char*)` | Dizgi olarak verilen metni `long` tipine dönüştürür ve geri döndürür. |
| `double strtod(const char*, char**)` | Dizgi olarak verilen metni `double` tipine dönüştürür ve geri döndürür. İkinci parametre ise metinde kayan noktalı sayıya çevrilmeyen ilk karakterin göstericisidir. |
| `long strtol(const char*, char**, int)` | Dizgi olarak verilen metni `long` tipine dönüştürür ve geri döndürür. İkinci parametre ise metinde kayan noktalı sayıya çevrilmeyen ilk karakterin göstericisidir. Üçüncü parametre ise metnin hangi sayı tabanında sayı içerdiğidir. Bu 2, 8, 10, 16 olabilir. |

*Tablo 26. Metinden İlkel Veri Tiplerine Dönüşüm Fonksiyonları*

Aşağıda bu fonksiyonları içeren bir program örneği verilmiştir.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    char tamsayiMetin[]="-23234";
    char kayannoktalisayiMetin[]="+2.3234e-1";
    char metin[]="-12,3e-2'nin karesi";
    int i = atoi(tamsayiMetin);
    printf("Tamsayı: %d\n",i); // Tamsayı: -23234
    double d= atof(kayannoktalisayiMetin);
    printf("Kayan Noktalı Sayı: %lf\n",d);
                               // Kayan Noktalı Sayı: 0.232340
    double dm;
    char* gerikalanmetin;
    dm= strtod(metin,&gerikalanmetin);
    printf("Metinden Alınan Sayı: %lf\n",dm);
                               // Metinden Alınan Sayı: -12.000000
    printf("Geri Kalan Metin: %s\n",gerikalanmetin);
                               // Geri Kalan Metin: ,3e-2'nin karesi
    return 0;
}
```

# Arama Tabloları

Arama tabloları (lookup tables-LUT), önceden hesaplanmış belirli değerlerle doldurulmuş dizilerdir. Uzun iç içe `if`, `if-else` veya `switch` ifadeleri yerine, bir programının verimliliğini artırmak için bu tablolar kullanılabilir.

```
int kareler[10] = {1, 4, 9, 16, 25, 36, 49, 64, 81, 100};
float karekokler[5] = {1.0, 1.41421, 1.7320, 2.0, 2.2360};
```

Bu tablolarda her seferinde hesaplama yapmak yerine önceden hesaplanmış değerleri tutarız. Arama tabloları genellikle dizgilerden oluşur.

```
char* renkler[] ={"Beyaz", "Mavi", "Yeşil", "Kırmızı", "Siyah"};
```

Aşağıda örnek bir program verilmiştir;

```
#include <stdio.h>
enum renlerenum {BEYAZ,MAVI,YESIL,KIRMIZI,SIYAH};
int main() {
    int kareler[10] = {1, 4, 9, 16, 25, 36, 49, 64, 81, 100};
    for (int i = 0; i < 10; i++){
        printf("%d \tKaresi(%d): %d\n", i+1, i+1, kareler[i]);
    }
    float karekokler[5] = {1.0, 1.41421, 1.7320, 2.0, 2.2360};
    for (int i = 0; i < 5; i++){
        printf("%d \tKarekökü(%d): %0.4f\n", i+1, i+1, karekokler[i]);
    }
    char* renkler[] ={"Beyaz", "Mavi", "Yeşil", "Kırmızı", "Siyah"};
    int secim=YESIL;
    printf("Seçtiğiniz renk: %s",renkler[YESIL]);
    return 0;
}
/*Program Çıktısı:
1       Karesi(1): 1
2       Karesi(2): 4
3       Karesi(3): 9
4       Karesi(4): 16
5       Karesi(5): 25
6       Karesi(6): 36
7       Karesi(7): 49
8       Karesi(8): 64
9       Karesi(9): 81
10      Karesi(10): 100
1       Karekökü(1): 1.0000
2       Karekökü(2): 1.4142
3       Karekökü(3): 1.7320
4       Karekökü(4): 2.0000
5       Karekökü(5): 2.2360
Seçtiğiniz renk: Yeşil

...Program finished with exit code 0
*/
```