Bölüm 8 – Karakterler ve Stringler

İçerik

8.1 Giriş

Standart kütüphane fonksiyonları;

- String ve karakter işleminde kolaylık sağlarlar
- Programlar; karakterleri, stringleri, teks satırlarını ve bellek bloklarını işleyebilirler

Bu tekniklerin kullanım alanları;

- Kelime işlemleri
- Sayfa düzeni yazılımı
- Yazma programları

8.2 Karakter ve Stringlerin Temelleri

Karakterler

- Program blokları oluşturma
  - Her program anlamlı gruplandırılmış karakterlerin bir dizisidir
- Karakter sabitleri
  - Tek tırnak içinde bir karakterden oluşan bir **`int`** değerdir
  - Örneğin **`'z'`** , **`z`** nin tamsayı değerini temsil eder

Stringler (Katarlar)

- Tek bir birim olarak algılanan bir karakterler serisi
  - harfler, rakamlar ve özel karakterler (*, **/**, **$**) içerebilir
- String cümlesi (string sabiti) – çift tırnak içinde yazılır
  - **`"Merhaba"`** gibi.
- Stringler karakterlerin bir dizisidir
  - String ilk karaktere bir pointer-dır
  - String değeri ilk karakterin adresidir

String deklarasyonu

- Bir karakter dizisi veya **char *** tipi bir değişken olarak tanımlanır

```
char renk[] = "mavi";
char *renkPtr = "mavi";
```

- Karakter dizisi olarak tanımlanan bir string **'\0'** ile biter
- **renk 5** elemana sahiptir

Stringleri okuma

- **Scanf** kullanımı

```
scanf("%s", kelime);
```

- Girdiyi **kelime[]** dizisine kopyalar
- **&** işaretine gerek yok (string bir pointer-dır)
- Dizide '**\0**' için yer açmayı unutmayınız

- Karakter kullanım kütüphanesi
  - Karakter verilerinin kullanımında ve testinde kullanışlı fonksiyonlar içerir
  - Her fonksiyon argüment olarak bir karakter (veya **`int`**) veya **`EOF`** alır
    - EOF(end of file - ^z (Ctrl+z))
- Aşağıda **`<ctype.h>`** deki tüm fonksiyonlar verilmiştir

8.3 Karakter Kullanım Kütüphanesi

Prototip	Açıklama
`int isdigit( int c )`	Eğer **c** bir rakam ise **true** diğer durumlarda **false** gönderir.
`int isalpha( int c )`	Eğer **c** bir harf ise **true** diğer durumlarda **false** gönderir.
`int isalnum( int c )`	Eğer **c** bir rakam veya harf ise **true** diğer durumlarda **false** gönderir.
`int isxdigit( int c )`	Eğer **c** bir hexadesimal karakter ise **true** diğer durumlarda **false** gönderir.
`int islower( int c )`	Eğer **c** bir küçük harf ise **true** diğer durumlarda **false** gönderir.
`int isupper( int c )`	Eğer **c** bir büyük harf ise **true** diğer durumlarda **false** gönderir..
`int tolower( int c )`	Eğer **c** bir büyük harf ise **küçük harfini** diğer durumlarda **kendini** gönderir.
`int toupper( int c )`	Eğer **c** bir küçük harf ise **büyük harfini** diğer durumlarda **kendini** gönderir..
`int isspace( int c )`	Eğer **c** bir white-space karakter ise - yenisatır (`'\n'`), boşluk (`' '`), yeni sayfa (`'\f'`), yatay tab (`'\t'`), dikey tab(`'\v'`) - **true** aksi halde **false** gönderir.
`int iscntrl( int c )`	Eğer **c** bir kontrol karakteri ise **true** aksi halde **false** gönderir.
`int ispunct( int c )`	Eğer **c** boşluk, rakam veya harf dışında yazılabilir bir karakter ise **true** aksi halde **false** gönderir.
`int isprint( int c )`	Eğer **c** boşluk (`' '`) dahil yazılabilir bir karakter ise **true** aksi halde **false** gönderir.
`int isgraph( int c )`	Eğer **c** boşluk (`' '`) dışında yazılabilir bir karakter ise **true** aksi halde **false** gönderir.

```c
/* Fig. 8.2: fig08 02.c
   isdigit, isalpha, isalnum, ve isxdigit fonksiyonlarını kullanma */
#include <stdio.h>
#include <ctype.h>

int main()
{
   printf( "%s\n%s%s\n%s%s\n\n", "isdigit e göre: ",
       isdigit( '8' ) ? "8 = " : "8 =! ", " rakam",
       isdigit( '#' ) ? "# = " :
       "# =! ", "rakam" );
   printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n\n",
       "isalpha:",
       isalpha( 'A' ) ? "A = " : "A =! ", "harf",
       isalpha( 'b' ) ? "b = " : " b=! ", "harf",
       isalpha( '&' ) ? "& = " : "& =! ", "harf",
       isalpha( '4' ) ? "4 = " :
       "4 =! ", "harf" );
   printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
       "isalnum a göre :",
       isalnum( 'A' ) ? "A = " : "A =! ",
       "rakam veya harf",
       isalnum( '8' ) ? "8 = " : "8 =! ",
       "rakam veya harf",
       isalnum( '#' ) ? "# = " : "# =! ",
       "rakam veya harf");
   printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n%s%s\n",
       "isxdigit e göre :",
       isxdigit( 'F' ) ? "F = " : "=! ",
       "hexadecimal rakam",
       isxdigit( 'J' ) ? "J = " : " =! ",
       "hexadecimal rakam",
```

8.3 Karakter Kullanım Kütüphanesi

```
33          isxdigit( '7' ) ? "7 = " : "7 =! ",
34          "hexadecimal rakam",
35          isxdigit( '$' ) ? "$ = " : "$ =! ",
36          "hexadecimal rakam",
37          isxdigit( 'f' ) ? "f = " : "f =! ",
38          "hexadecimal rakam" );
39      return 0;
40  }
```

```
Isdigit e göre:
8 = rakam
# =! rakam

Isalpha ya göre:
A = harf
b = harf
& =! harf
4 =! harf

Isalnum a göre:
A = rakam veya harf
8 = rakam veya harf
# =! rakam veya harf

Isxdigit e göre:
F = hexadecimal rakam
J =! hexadecimal rakam
7 = hexadecimal rakam
$ =! hexadecimal rakam
f = hexadecimal rakam
```

8.4 String Dönüşüm Fonksiyonları

- Dönüşüm fonksiyonları
  - **<stdlib.h>** dedir (genel kullanım kütüphanesi).
- Rakam stringlerini tamsayı ve reel sayıya dönüştürür

Prototip	Açıklama
`double atof( const char *nPtr )`	**nPtr** stringini **double** a dönüştürür
`int atoi( const char *nPtr )`	**nPtr** stringini **int** e dönüştürür
`long atol( const char *nPtr )`	**nPtr** stringini long **int** e dönüştürür
`double strtod( const char *nPtr, char **endPtr )`	**nPtr** stringini **double** a dönüştürür
`long strtol( const char *nPtr, char **endPtr, int base )`	**nPtr** stringini **long** a dönüştürür
`unsigned long strtoul( const char *nPtr, char **endPtr, int base )`	**nPtr** stringini **unsigned long** a dönüştürür

8.4 String Dönüşüm Fonksiyonları

```
1  /* Fig. 8.6: fig08_06.c
2   atof kullanımı*/
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main()
7  {
8     double d;
9
10    d = atof( "99.0" );
11    printf( "%s%.3f\n%s%.3f\n",
12            "string = \"99.0\" dönüştürülen double =", d,
13            "Dönüşen değerin yarısı =  ",
14            d / 2.0 );
15    return 0;
16 }
```

```
string = "99.0" dönüştürülen double = 99.000
Dönüştürülen değerin yarısı = 49.500
```

8.5 Standart Input/Output Kütüphane Fonksiyonları

- Fonksiyonlar **<stdio.h>** dedir
  - Karakter ve string verileri için kullanılır

Fonksiyon prototipi	Açıklama
`int getchar( void );`	Standart girdi aygıtından bir sonraki karakteri alır ve onu bir tamsayı olarak gönderir
`char *gets( char *s );`	Standart girdi aygıtından karakterleri alır ve yeni satır veya EOF karakteri girilinceye kadar **s** dizisine aktarır Diziye en son boşluk karakteri eklenir
`int putchar( int c );`	**c** de tutulan karakteri yazar
`int puts( const char *s );`	Yeni satırda **s** stringini yazar.
`int sprintf( char *s, const char *format, ... );`	**printf** -e denktir Tek farkı çıktı ekran yerine **s** dizisine yüklenir.
`int sscanf( char *s, const char *format, ... );`	**scanf** -e denktir . Tek farkı çıktı klavye yerine **s** dizisinden yüklenir.

```c
/* Fig. 8.13: fig08_13.c
   gets ve putchar kullanımı */
#include <stdio.h>

int main()
{
   char cumle[ 80 ];
   void ters( const char * const );

   printf( "Bir satır yaz:\n" );
   gets( cumle );

   printf( "\n Tersden yazım::\n" );
   ters( cumle );

   return 0;
}

void ters( const char * const sPtr )
{
   if ( sPtr[ 0 ] == '\0' )
      return;
   else {
      ters( &sPtr[ 1 ] );
      putchar( sPtr[ 0 ] );
   }
}
```

ters orjinal stringin alt stringini kullanarak kendini çağırır.
'\0' karakterine ulaştığında **putchar** kullanarak yazar

```
Bir satır yaz:
Karakter ve stringler

Tersden yazım::
relgnirts ev retkaraK
```

- String kullanım kütüphanesi fonksiyonları;
  - String verilerini işler
  - Stringleri tarar
  - Stringleri fişler
  - String uzunluğunu belirler

Prototip	Açıklama
`char *strcpy( char *s1, const char *s2 )`	**s2** stringini **s1** dizisine kopyalar. **s1** değerini gönderir.
`char *strncpy( char *s1, const char *s2, size_t n )`	**s2** stringinin en fazla **n** karakterini **s1** dizisine kopyalar. **s1** değerini gönderir..
`char *strcat( char *s1, const char *s2 )`	**s2** stringini **s1** dizisine ekler. **s2** nin ilk karakeri **s1** in boşluk karakteri üzerine yazılır. Yeni **s1** değerini gönderir.
`char *strncat( char *s1, const char *s2, size_t n )`	**s2** stringininin en fazla **n** karakterini **s1** dizisine ekler. **s2** nin ilk karakeri **s1** in boşluk karakteri üzerine yazılır. Yeni **s1** değerini gönderir.

8.6 String Kullanım Fonksiyonları

```
1  /* Fig. 8.19: fig08_19.c
2     strcat ve strncat kullanımı */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     char s1[ 20 ] = "Mutlu ";
9     char s2[] = "Yıllar ";
10    char s3[ 40 ] = "";
11
12    printf( "s1 = %s\ns2 = %s\n", s1, s2 );
13    printf( "strcat( s1, s2 ) = %s\n", strcat( s1, s2 ) );
14    printf( "strncat( s3, s1, 6 ) = %s\n", strncat( s3, s1, 6 ) );
15    printf( "strcat( s3, s1 ) = %s\n", strcat( s3, s1 ) );
16    return 0;
17 }
```

```
s1 = Mutlu
s2 = Yıllar
strcat( s1, s2 ) = Mutlu Yıllar
strncat( s3, s1, 6 ) = Mutlu
strcat( s3, s1 ) = Mutlu Mutlu Yıllar
```

- Stringleri karşılaştırma
  - Bilgisayar stringdeki karakterlerin nümerik ASCII kodlarını karşılaştırır
  - EK D de karakter kodlarının listesi verilmiştir

```
int strcmp( const char *s1, const char *s2 );
```

  - **s1** ve **s2** stringlerini karşılaştırır
  - Eğer **s1 < s2** ise bir negatif sayı, **s1 == s2** ise sıfır ve **s1 > s2** ise bir pozitif sayı gönderir.

```
int strncmp( const char *s1, const char *s2,
   size_t n );
```

  - **s1** ve **s2** stringlerininin ilk **n** karakterini karşılaştırır.
  - Yukarıdaki gibi değer gönderir.

8.8 String Araştırma (Search) Fonksiyonları

Prototip	Açıklama
`char *strchr( const char *s, int c );`	**s** stringindeki **c** karakterinin ilk göründüğü yeri bulur. **c** bulunursa, s deki c ye bir pointer gönderilir. Aksi halde, **NULL** pointer gönderilir.
`size_t strcspn( const char *s1, const char *s2 );`	**s1** deki **s2** de olmayan ilk karakter bloğunun uzunluğunu belirler ve gönderir.
`size_t strspn( const char *s1, const char *s2 );`	**s1** deki **s2** de de olan ilk karakter bloğunun uzunluğunu belirler ve gönderir
`char *strpbrk( const char *s1, const char *s2 );`	**s1** in herhangi bir karakterinin **s2** de ilk göründüğü yeri belirler. **s2** de bu karakter bulunursa, karakterin **s1** deki yerine bir pointer gönderilir,. Aksi halde, **NULL** pointer gönderilir.
`char *strrchr( const char *s, int c );`	**s** de **c** nin son göründüğü yeri belirler. **c** bulunursa, **c** ye bir pointer gönderilir. Aksi halde, **NULL** pointer gönderilir.
`char *strstr( const char *s1, const char *s2 );`	**s1** stringinde **s2** stringinin ilk göründüğü yeri bulur. Eğer var ise, o noktaya bir pointer gönderilir. Aksi halde, **NULL** pointer gönderilir.
`char *strtok( char *s1, const char *s2 );`	**strtok** un bir çağrımlar dizisi **s1** stringini kelimeler gibi mantıksal parçalara bölüp, **s2** stringindeki karakter ile karşılaşıldığında yeni parça oluşturur. İlk çağrı, ilk argüment olarak **s1** i içerir. Sonraki çağrılar **NULL** gelene kadar **s1** i parçalar. Her çağrımda o anki parçaya bir pointer gönderilir.

8.8 String Araştırma (Search) Fonksiyonla

```
1  /* Fig. 8.27: fig08_27.c
2     strspn kullanımı*/
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     const char *string1 = "Değer 3.14159 olur";
9     const char *string2 = "rDeğ. 3";
10
11    printf( "%s%s\n%s%s\n\n%s\n%s%u\n",
12            "string1 = ", string1, "string2 = ", string2,
13            "string1 in sadece string2 den değerler içeren",
14            "başlangıç bloğunun uzunluğu = ",
15            strspn( string1, string2 ) );
16    return 0;
17 }
```

```
string1 = Değer 3.14159 olur
string2 = rDeğ. 3

String1 in sadece string2 den değerler içeren
başlangıç bloğunun uzunluğu = 8
```

8.8 String Araştırma (Search) Fonksiyonla

```c
/* Fig. 8.29: fig08 29.c
 strtok kullanımı*/
#include <stdio.h>
#include <string.h>

int main()
{
   char string[] = "Bu 5 kelimeli bir cümledir";
   char *tokenPtr;

   printf( "%s\n%s\n\n%s\n",
           "Parçalanacak string:", string,
           "Parçalar:" );

   tokenPtr = strtok( string, " " );

   while ( tokenPtr != NULL ) {
      printf( "%s\n", tokenPtr );
      tokenPtr = strtok( NULL, " " );
   }

   return 0;
}
```

```
Parçalanacak string:
Bu 5 kelimeli bir cümledir

Parçalar:
Bu
5
kelimeli
bir
cümledir
```

8.9 String Bellek Fonksiyonları

- Bellek Fonksiyonları
  - **<stdlib.h>** dedir
  - Bellek bloklarını araştırır bulur ve karşılaştırır
  - Her hangi veri bloğunu düzenler
- Pointer parametreleri **void *** dır
  - **void *** a herhangi bir pointer atanabilir veya tersi.
  - **void *** a tekrar referans verilemez
    - Her bir fonksiyon, işlenecek byte (karakter) sayısını belirleyen bir boyut argumenti alır

8.9 String Bellek Fonksiyonları

Prototip	Açıklama
`void *memcpy( void *s1, const void *s2, size_t n )`	**s2** ile işaret edilen nesnenin **n** karakterini **s1** ile işaret edilen nesneye kopyalar. Elde edilen nesneyi işaret eden bir pointer gönderilir.
`void *memmove( void *s1, const void *s2, size_t n )`	**s2** ile işaret edilen nesnenin **n** karakterini **s1** ile işaret edilen nesneye kopyalar. Elde edilen nesneyi işaret eden bir pointer gönderilir. Kopyalama işlemi önce **s2** nin yerinden bir geçici diziye ve sonra da geçici diziden s1 in yerine şeklinde yapılır. Elde edilen nesneyi işaret eden bir pointer gönderilir.
`int memcmp( const void *s1, const void *s2, size_t n )`	**s1** ve **s2** ile işaret edilen nesnelerin ilk **n** karakterlerini karşılaştırır. Fonksiyon, **s1 s2** ye eşit, küçük eşit veya büyük eşit ise sırası ile 0 , negatif veya pozitif değer gönderir
`void *memchr(const void *s, int c, size_t n )`	**s** ile işaretlenen nesnenin ilk n karakterinde, **c** nin ilk göründüğü yeri belirler (**unsigned char** a dönüştürerek). Eğer **c** bulunursa, nesnedeki **c** ye bir pointer gönderilir. Aksi halde **0** gönderilir.
`void *memset( void *s, int c, size_t n )`	**s** ile işaretlenen nesnenin ilk n karakterine, **c** (**unsigned char** a dönüştürür) yi kopyalar. Elde edilene bir pointer gönderir.

8.9 String Bellek Fonksiyonları

```
1  /* Fig. 8.32: fig08_32.c
2   memmove kullanımı*/
3  #include <stdio.h>
4  #include <string.h>
5
6  int main()
7  {
8     char x[] = "Evim Güzel Evim";
9
10    printf( "%s%s\n",
11            " memmove dan önce x dizisindeki string: ", x );
12    printf( "%s%s\n",
13            " memmove dan sonra x dizisindeki string :  ",
14            memmove( x, &x[ 5 ], 10 ) );
15
16    return 0;
17 }
```

```
memmove dan önce x dizisindeki string: Evim Güzel Evim
memmove dan sonra x dizisindeki string: Güzel Evim Evim
```

8.10 Diğer String Fonksiyonları

- **`char *strerror( int hatano );`**
  - **`hatano`** ya bağlı olarak sisteme-bağlı bir hata gönderir
  - Stringe bir pointer gönderir
- **`size_t strlen( const char *s );`**
  - **`s`** stringindeki (**`NULL`** dan önceki) karakter sayısını gönderir

8.10 Diğer String Fonksiyonları

```c
/* Fig. 8.37: fig08_37.c
 strerror kullanımı*/
#include <stdio.h>
#include <string.h>

int main()
{
   printf( "%s\n", strerror( 2 ) );
   return 0;
}
```

```
No such file or directory
```