BÖLÜM 5: TEMEL GİRİŞ/ÇIKIŞ FONKSİYONLARI

Bu bölümde, C programlama dilinde kullanılan temel giriş ve çıkış fonksiyonları ele alınacaktır. C programlama dilinde default (varsayılan) giriş cihazı klavye (keyboard), default çıkış cihazı ise ekrandır.

I. FORMATLI GİRİŞ VE ÇIKIŞ FONKSİYONLARI

`printf` fonksiyonu: `printf` fonksiyonu, formatlı çıkış fonksiyonudur. `printf` fonksiyonu kullanılarak ekranda görüntülenecek olan değişken, ifade veya sabit, bir format string'i ile gönderilir. Şimdi aşağıdaki programı ele alalım.

```
#include <stdio.h>
void main()
{
    printf("Merhaba\n");
}
```

Program çıktısı:
```
Merhaba
```

`printf` fonksiyonunun `stdio.h` header dosyasında yer aldığını daha önce belirtmiştik. Bu programda, `printf` fonksiyonu kullanıldığı için, `stdio.h` dosyası programa dahil edilmiştir. Program çalıştırıldığında ekranda; `Merhaba` yazısı görüntülenecektir. `\n` kullanılarak kursörün, `Merhaba` yazısından sonra bir alt satıra inmesi sağlanmıştır.

`#include` direktifi programın en başında yer alır. Daha sonra, `main()` yazılarak ana program başlatılır. `main()`, tüm C programlarında bulunması gereken ana fonksiyonun adıdır. `main()`'den herhangi bir değer döndürülmediği için `void` ifadesi eklenir. `main()`'den sonra ise küme parantezi açılır (`{`) ve değişken tanımlamaları ile fonksiyonlar ve işlemler sıralanır. Programın en sonunda da küme parantezi kapatılır (`}`).

```
#include <stdio.h>
void main()
{   int x, y;
    x = 10;
    y = 20;
    printf("x = %d y = %d\n", x, y);
}
```

Program çıktısı:
```
X = 10 y = 20
```

Bu programda ise, `int x, y` satırı kullanılarak `x` ve `y` adında integer (tamsayı) tipi iki değişken tanımlanmış ve `x = 10` yazılarak x değişkenine 10 sayısı aktarılmış, `y = 20` yazılarak y değişkenine 20 sayısı aktarılmıştır. Daha sonra `printf` fonksiyonu kullanılarak `x` değişkeninin içindeki 10 sayısı ve `y` değişkeninin içindeki 20 ekrana yazdırılmıştır. `printf` fonksiyonunda `"` (çift

tırnak) içersinde verilen bölüm format stringi olarak adlandırılır ve çıktının hangi formatta olacağını belirler. Bu programda çıkış yapılan sayılar integer (tamsayı) olduğu için `%d` formatı kullanılmıştır. Daha sonra ise, format stringi kullanılarak ekrana yazdırılacak olan değişken veya değişkenler yer alır. Format stringi ile çıkışa aktarılacak ilk değişken arasında (`,`) karakteri kullanılır. Birden fazla değişken veya ifade çıkışa aktarılacaksa, bunlar birbirinden (`,`) ile ayrılır.

C programlama dilinde, işlemler birbirlerinden (`;`) ile ayrılır. İşlemleri ayrı satırlarda yazmak koşul değildir, bir satırda birden fazla işlem yazılabilir, ancak bu işlemlerin birbirinden (`;`) ile ayrılması gerekir. Programların anlaşılabilir olması açısından, işlemlerin ayrı satırlara yazılması ve bloklar arasında boşluk bırakılması yolu benimsenmektedir. Ancak bu zorunlu değildir, sadece programı daha anlaşılır kılmak amacına yöneliktir. Daha önce de belirtildiği gibi, C programlama dilinde program yazarken sadece küçük harfler kullanılmaktadır. Bu kuralın birkaç istisnası vardır ve bunlar da daha önce açıklanmıştı.

`printf` fonksiyonunda format stringi kullanılarak çıkışın formatı tanımlanmaktadır. Aşağıdaki listede çeşitli değişken tiplerinin çıkışlarında kullanılması gereken format tanımlayıcılar görülmektedir. Bu format tanımlayıcılar, daha sonra göreceğimiz, formatlı giriş fonksiyonu olan `scanf` fonksiyonu ile de kullanılmaktadır.

Tip Karakteri	Anlamı
`%c`	tek bir karakter giriş veya çıkışı
`%d`	işaretli ondalık tamsayı giriş veya çıkışı
`%ld`	uzun işaretli ondalık tamsayı giriş veya çıkışı
`%u`	işaretsiz ondalık tamsayı giriş veya çıkışı
`%f`	gerçel sayı giriş veya çıkışı
`%e veya %E`	Gerçel sayı girişi veya çıkışı (üstlü)
`%s`	karakter dizisi (string) giriş veya çıkışı
`%lu`	işaretsiz uzun tamsayı giriş veya çıkışı
`%o`	oktal sayı giriş veya çıkışı
`%x`	hexadesimal sayı giriş veya çıkışı
`%p`	pointer (işaretçi) tipi değişken giriş veya çıkışı

Bir programda çıktılar yazıcıdan (printer) alınmak isteniyorsa `fprintf` fonksiyonu kullanılır. Aşağıdaki program çalıştırıldığında çıktı yazıcıdan alınacaktır.

```
#include <stdio.h>
void main()
{
    fprintf(stdprn, "Pamukkale...\n");
}
```

Programın printer çıktısı:

```
Pamukkale...
```

Bu programda çıktının yazıcıdan alınacağı `stdprn` kodu ile belirtilmiştir. Program çalıştırılırken, yazıcı çıkışa hazır hale getirilmelidir, aksi halde program hata mesajı verecektir.

Bazı programlarda, işlenecek olan bilgiler programın içinde tanımlanır ve klavyeden herhangi bir giriş yapılmaz. Bu tip programlara "non-interactive" programlar denir. Aşağıdaki programda iki sayı toplanmakta ve bu sayılar ve toplamları görüntülenmektedir.

```
#include <stdio.h>
void main()
{   int a,b,c;
    a = 8;
    b = 5;
    c = a + b;
    printf("a      = %8d\n", a);
    printf("b      = %8d\n", b);
    printf("Toplam = %8d\n", c);
}
```

Program çıktısı:

```
a      =     8
b      =     5
Toplam =    13
```

Bu programda `a`, `b` ve `c` değişkenleri integer (tamsayı) cinsinden tanımlanmıştır. `a=8` işlemiyle `a` değişkenine 8 sayısı, `b=5` işlemiyle de `b` değişkenine 5 sayısı aktarılır. `c=a+b` işlemiyle de `a` ve `b` değişkenlerinin tuttuğu sayılar toplanmakta ve sonuç `c` değişkenine aktarılmaktadır. `a` ve `b` değişkenlerinin değerleri 8 basamak sağa yanaşık, `c` değişkeninin değeri ise yine 8 basamak sağa yanaşık olarak sola doğru kaydırılarak ekran yazdırılmaktadır. Aşağıdaki şekilde, bu programın çalıştırılması sonucunda `a`, `b` ve `c` değişkenlerinin aldığı bellek değerleri görülüyor.

a	8
b	5
c	13

Değişkenler, bilgisayar belleğinde bilgilerin depo edildiği kutular olarak düşünülebilir. Bu kutuların ise işletim sistemleri tarafından atanan adresleri vardır. Değişken adı, bilginin saklandığı kutunun adı olmaktadır. Kutunun içinde ise bilgi bulunur. Bu bilgiye değişkenin adı kullanılarak ulaşılabilir. Yine bir kutunun içine, değişken adı kullanılarak bilgi yazılabilir.

Bir değişkene iki ayrı işlemle iki ayrı sayı aktarılırsa ikinci aktarılan geçerli olur yani ilk bilgi silinir.

```
a = 15;
a = 20;
```

Yukarıdaki işlemlerin bir programda çalıştırılması sonucu önce `a` değişkenine 15 sayısı aktarılır. Daha sonra `a=20` işlemi, `a` değişkenine 20 sayısını aktaracak ve bu sırada 15 sayısı silinmiş olacaktır.

Şimdi de bölme işlemi yapan programı görelim.

```
#include <stdio.h>
void main()
{   float a,b,c;
    a = 25;
    b = 4;
    c = a/b;
    printf("a=      %5.2f\n", a);
    printf("b=      %5.2f\n", b);
    printf("Bolum=  %5.2f\n", c);
}
```

Program çıktısı:

```
a=      25.00
b=       4.00
Bolum=   6.25
```

Bu programda `a`, `b` ve `c` değişkenleri `float` (kayan noktalı) sayı tipinde tanımlanmıştır. `(/)` operatörü kullanılarak `a` değişkeninin tuttuğu 25 sayısı, `b` değişkeninin tuttuğu 4 sayısına bölünmüş ve sonuç `c` değişkenine aktarılmıştır.

Format stringi tanımında `%5.2f` kullanılmıştır. `float` tipi sayıların çıkış tanımlamalarında, sayının tamsayı ve kesirli bölümünün kaç basamaklı olarak görüntüleneceği, programcı tarafından tanımlanabilmektedir. `%5.2f` ifadesindeki `5`, float tipi sayının toplam dijit sayısını (`.` dahil) `2` ise noktadan sonraki dijitlerin sayısını belirtmektedir. Bu durum aşağıdaki şekilde ifade edilir.

`a` sayısı	2	5	.	0	0
`b` sayısı		4	.	0	0
`c` sayısı		6	.	2	5

Çıkış formatının tanımında, sadece `%f` format tanımlayıcısı kullanılırsa yani `float` tipi sayının tamsayı ve kesirli kısımlarının basamak uzunlukları verilmezse, görüntülenen `float` tipi sayının tamsayı kısmı olduğu gibi, kesirli kısmı ise 6 basamaklı olarak görüntülenir. Bu durumda birden fazla sayının alt alta görüntülenmesinde kesirli kısmı ayıran noktalar alt alta gelmeyecektir.

Yukarıdaki programda format stringte `%5.2f` format tanımlayıcısı kullanılarak, görüntülenen float tipi üç sayının kesir noktaları alt alta getirilmiştir. Şüphesiz, bu durum daha iyi bir görüntü elde etmemizi sağlamakta ve herhangi bir karışıklığa meydan vermemektedir.

C programlama dilinde, diğer programlama dillerinin tersine, aynı işlemde birden fazla değişkene aynı değeri aktarmak mümkündür. Aşağıdaki programda `x` ve `y` değişkenlerine aynı işlemde değer atanmıştır.

```
#include <stdio.h>
void main()
{   int x, y;
    x = y = 25;
    printf("x= %d\n",x);
    printf("y= %d\n",y);
}
```

Program çıktısı:
```
x= 25
y= 25
```

Yukarıdaki programda `x=y=25` işlemiyle, 25 sayısı önce `y` değişkenine, daha sonra 25 olan `y`'nin değeri `x` değişkenine aktarılmıştır.

Programların yazılması sırasında bazı açıklamalar yapma gereği ortaya çıkabilir. Bu açıklamalar program metninin, okuyan kişiler tarafından daha kolay anlaşılmasını sağlar. C programlama dilinde, programla ilgili açıklama ve yorumlar `/*...*/` karakterleri arasında kalan bölümde verilir. Bu açıklamalar, programın derlenmesi sırasında derleyici (compiler) tarafından dikkate alınmaz. Açıklamalar, `/* */` karakterleri arasında olmak koşuluyla, programın herhangi bir yerinde yer alabilir. Tek satırlık açıklamalarda ise genellikle çift bölü "`//`" tercih edilmektedir. Bu durumda çift bölü ile başlayan satır derleme sırasında dikkate alınmaz. Aşağıda, bazı açıklama örnekleri görülmektedir.

```
/*Bu program ortalama hesaplar*/

//Bu program ortalama hesaplar

/* Sıralama programı
Yazan A. Kadir YALDIR
10.10—2003 */
```

Üçüncü açıklama şeklinde, açıklama 3 satırda yer almaktadır. Ancak açıklamanın en başında `/*` karakteri, en sonunda ise `*/` karakteri yer almıştır.

C programlama dilinde, başka programlama dillerinde bulunmayan unary operatörler olduğunu daha önce açıklamıştık. Unary operatörler (`++` ve `--`), değişkenlerin tuttuğu sayıları 1 arttırır veya 1 azaltır. Aşağıdaki programda unary operatörler kullanılmıştır. Ayrıca, programın çeşitli yerlerinde bazı açıklamalar yapılmaktadır.

```
/*Bu programda unary operatörler kullanılmaktadır*/
#include <stdio.h>
void main()
{   int x, y; /*değişken tanımlama*/
    x = 5;
    y = 5;
    printf("++x değeri= %d\n",++x); /*önce artır sonra yaz*/
    printf("y++ değeri= %d\n",y++); /*önce yaz sonra artır*/
    printf("\n"); /*bir satır boşluk*/
    printf("Artırma işleminden sonra x= %d\n",x);
    printf("Artırma işleminden sonra y= %d\n",y);
}
```
Program çıktısı:
```
++x değeri= 6
y++ değeri= 5

Artırma işleminden sonra x= 6
Artırma işleminden sonra y= 6
```

Yukarıdaki programda ++ unary operatörü kullanılmıştır. Bu operatör, değişkenin tuttuğu sayının değerini 1 arttırır. Ancak, ++ operatörünün değişkenin ön veya arka tarafında kullanılması özellikle bu değişkenin bir ifade veya işlemde kullanılması durumunda değişiklik göstermektedir.

```
x=5;
y=x++;  .
```

Yukarıdaki satırlar çalıştırıldığında x değişkenine aktarılan 5 sayısı önce y değişkenine aktarılır (x'in değeri olarak), daha sonra x'in değeri 1 arttırılır. Yani yukarıdaki satırların çalıştırılması sonucunda y değişkeninde 5, x değişkeninde ise 6 sayısı bulunur.

```
x = 5;
y = ++x;
```

satırlarının çalıştırılması sonunda ise x değişkeninin tuttuğu 5 sayısı önce 1 arttırılır, daha sonra ise bu artmış şekliyle y değişkenine aktarılır. Bu satırların çalıştırılması sonucunda x ve y değişkenlerinin her ikisinde de 6 sayısı bulunur.

```
x = 10;
y = 10;
printf("x = %d\n",x--);
printf("y = %d\n",--y);
```

Yukarıdaki satırlar çalıştırıldığında ise x değişkeni önce görüntülenir, daha sonra içindeki değeri 1 azaltılır. y değişkeni ise, değeri 1 azaltıldıktan sonra görüntülenir ve

```
x = 10
y = 9
```

sonucu elde edilir.

Aşağıdaki programda % (modül) operatörü kullanılarak, bölme işlemi sonucunda kalan sayı elde edilmiştir.

```
/* Bu programda modül operatörü kullanılmıştır */
#include <stdio.h>
void main()
{   int x, y, z;
    x = 15;
    y = 4;
    z= x % y;
    /*Bölme işlemi sonucunda "kalan" sayı bulunuyor*/
    printf("z= %d\n", z);
}
```

Program çıktısı:

```
z= 3
```

15 sayısı 4 sayısına bölündüğünde bölüm 3 kalan 3 elde edilir. Yukarıdaki programda, kalan 3 sayısı elde edilmektedir.

`scanf` fonksiyonu: Birçok programda, işlenecek bilgiler (veri) programın çalışması sırasında kullanıcı tarafından bilgisayara verilir. Bu tip programlara "interactive" programlar denir. `scanf` fonksiyonu klavyeden ENTER tuşuna basarak bilgi girişine yarar. Aşağıdaki program, klavyeden girilen iki sayıyı toplar ve sonucu görüntüler.

```
#include <stdio.h>
void main()
{   int a, b, c;
    printf ("Birinci sayiyi giriniz: ");
    scanf ("%d", &a);
    printf ("Ikinci sayiyi giriniz: ");
    scanf("%d", &b);
    c=a+b;
    printf("Sayilarin toplami= %d\n", c);
}
```

Program çıktısı:

```
Birinci sayiyi giriniz: 9
Ikinci sayiyi giriniz: 8
Sayilarin toplami= 17
```

Bu programda, `scanf` fonksiyonu kullanılarak klavyeden girilen birinci sayı `a` değişkenine, ikinci sayı ise `b` değişkenine atanır. `scanf` fonksiyonunda, `a` değişkenine atama yapılırken `&` (ampersand) adres operatörü kullanılır. `&a`, `a` değişkenin adresini temsil eder. `&b` ise, `b` değişkeninin adresini temsil etmektedir. `scanf` fonksiyonunda, `printf` fonksiyonunda olduğu gibi format stringi vardır. Yukarıdaki programda, klavyeden girilen sayılar integer tipi olduğu için "`%d`" format tanımlayıcısı kullanılmıştır.

Aşağıdaki program, klavyeden girilen iki integer sayının farkını bulur ve sonucu görüntüler.

```
#include <stdio.h>
void main()
{   int a, b, c;
    printf("Birinci sayiyi giriniz: ");
    scanf("%d", &a) ;
    printf("Ikinci sayiyi giriniz: ");
    scanf("%d", &b);
    c=a–b;
    printf("Fark = %d\n", c);
}
```

Program çıktısı:

```
Birinci sayıyı giriniz: 25
İkinci sayıyı giriniz: 15
Fark= 10
```

`scanf` fonksiyonu kullanılarak klavyeden yapılan girişlerde, girilen sayı yazıldıktan sonra mutlaka ENTER tuşuna basılması gerekir.

Aşağıdaki program, klavyeden yarıçap uzunluğu girilen dairenin alanını hesaplar.

```
/* Bu program yarıçapı klavyeden girilen
dairenin alanını hesaplar */

#include <stdio.h>
#define PI 3.14159
void main()
{   float r, a;
    printf("Dairenin yaricapini giriniz:");
    scanf("%f", &r);
    a=PI*r*r;
    printf("\n");
    printf("Dairenin alani:%8.3f\n", a);
}
```

Program çıktısı:

```
Dairenin yarıçapını giriniz:10

Dairenin alanı: 314.159
```

Bu programın başlangıcında, `#define` direktifi kullanılarak `PI` sabitinin değeri 3.14159 olarak tanımlanmıştır. Program içinde, `PI` sabiti, 3.14159 sayısını temsil eder.

Klavyeden iki farklı sayı okunmak istendiğinde `scanf()` fonksiyonu şöyle de kullanılabilir.

```
scanf("%d%f", &x, &y);
```

veriler

```
16 1.568 (Enter)

yada
16        1.568 (Enter)

veya
16 (Enter)
1.568 (Enter)
```

şeklinde girilebilir.

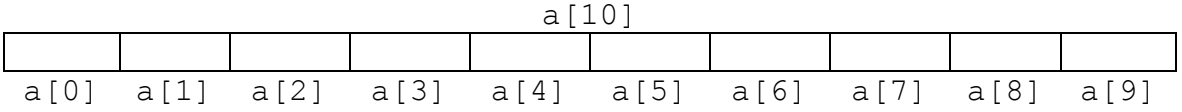
C programlama dilinde, string tipinde bir değişken tanımı yoktur. String, harflerin çeşitli kombinasyonlarıyla elde edilmiş sözcük veya cümle demektir. Örneğin bir kişinin adı, soyadı, bir şehir adı string ifadelerdir. C programlama dilinde string tipi ifadeleri tutan değişkenler `char` tipi dizi değişken olarak tanımlanır.

```
char a;
```

Yukarıdaki şekilde tanımlanan char tipi `a` değişkeni, sadece bir karakterlik bilgiyi tutabilir.

```
char a[10];
```

a değişkeni bu şekilde tanımlandığında ise bilgisayar belleğinde a değişkeni için 10 adet karakteri tutacak şekilde bir yer ayrılır. Bu tanımlama sonucunda, bilgisayar belleğinde ayrılan alan aşağıdaki gibi gösterilebilir.



Yukarıdaki alana, her bir kutuya bir harf olmak üzere 9 harfli bir sözcük yazılabilir. Son kutuya \0 (string sonu karakteri -NULL) derleyici tarafından yazılacağı için, string tipi değişken tanımlanırken, sözcükteki harf sayısının bir fazlası kadar yer tanımlamak gerekir. İlk kutunun a[0] olduğuna dikkat etmelisiniz. Buna göre ad[20] şeklinde bir değişken tanımlandığında ilk kutu ad[0], son kutu ise ad[19] olmak üzere değişken 20 kutudan oluşacaktır.

Aşağıdaki programda, klavyeden girilen isim ekranda görüntülenmektedir.

```
#include <stdio.h>
void main()
{   char ad[20];
    printf ("Adinizi giriniz: ");
    scanf("%s", ad) ;
    printf("\n");
    printf("Merhaba %s\n", ad);
}
```

Program çıktısı:

```
Adinizi giriniz: Ahmet

Merhaba Ahmet
```

Bu programda scanf fonksiyonu kullanılarak, klavyeden girilen isim ad değişkenine aktarılmaktadır. Klavyeden, string tipi bir ifade girildiği için, scanf fonksiyonunda format tanımlayıcı olarak %s kullanılmıştır. char ad[20] tanımıyla, bilgisayar belleğinde, ad değişkeni için 20 karakterlik bir alan ayrılır ve girilen isim yerleştirilir.

Klavyeden girilen isim ad[0] kutusundan başlamak üzere her kutuya bir harf olmak üzere yerleştirilir ve son harften sonra "\0" (NULL) yani string sonu karakteri konulur. Harflerin kutulara yerleştirilmesi ve string sonu karakterinin konulması işlemi program tarafından yapılır.

printf fonksiyonunda, format tanımlayıcı olarak "%s" kullanılarak görüntülenecek olan ifadenin string tipi olduğu gösterilmiştir. String tipi ifadelerin görüntülenmesinde, printf fonksiyonu ad[0] kutusundan başlamak üzere "\0" (string sonu) karakterini görünceye kadar bütün kutulardaki karakterleri görüntüler, böylece klavyeden girilen isim görüntülenmiş olur.

Aşağıdaki programda, `char` tipi tanımlanmış bir dizi değişkene, program içinden bir string ifade aktarılmaktadır.

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

void main()
{
    char ch[30];
    strcpy(ch, "MERHABA, NASILSINIZ...");
    printf("%s\n", ch);
}
```

Program çıktısı:
```
MERHABA, NASILSINIZ...
```

Bu programda, `ch[30]` adlı `char` tipi dizi değişkene, bir string ifade aktarılmıştır. Bu işlemi `strcpy` fonksiyonu yapar. `strcpy` fonksiyonu `string.h` header dosyasında bulunduğu için, programın başında `string.h` header dosyası programa dahil edilmiştir.

C programlama dilinde pointer tipi değişken tanımlamak mümkündür. Pointer, başka bir değişkenin adresini tutan değişkendir. Pointer konusu, derslerimizin ilerleyen bölümlerinde ayrıntılı olarak ele alınacaktır. Aşağıdaki programda, `ch` değişkeni pointer tipi tanımlanmıştır.

```
#include <stdio.h>

void main()
{
    char *ch;
    ch = "C PROGRAMLAMA...";
    printf("%s\n", ch);
}
```

Program çıktısı:
```
C PROGRAMLAMA...
```

`char *ch` satırıyla `ch` değişkeni pointer tipinde tanımlanmıştır. Bir değişkenin pointer olarak tanımlanması için değişken adının başına `*` (asteriks) işareti konulur. Daha sonra `ch = "C PROGRAMLAMA..."` satırıyla da, pointer tipi değişkenin işaret ettiği adrese `"C PROGRAMLAMA..."` stringi aktarılmıştır. Pointer, bu stringin ilk harfinin yerleştirildiği kutunun adresini gösterir. Stringin diğer harfleri de sırayla bitişik kutulara yerleştirilir ve son karakterin bulunduğu kutudan sonra gelen kutuya da "`\0`" (string sonu) karakteri konulur. `printf` fonksiyonu, `ch` tarafından gösterilen adresten başlamak üzere, "`\0`" karakterinin bulunduğu adrese kadar, tüm karakterleri görüntüler.

II. FORMATSIZ GİRİŞ VE ÇIKIŞ FONKSİYONLARI

C programlama dilinde kullanılan formatsız giriş ve çıkış fonksiyonları aşağıda verilmiştir.

Fonksiyon	İşlem
`getchar()`	Klavyeden bir karakter okur ve ENTER tuşuna basılmasını bekler.
`getche()`	Klavyeden bir karakter okur ve ENTER tuşuna basılmaksızın okunan karakteri bir değişkene aktarır. Klavyeden girilen karakter ekranda görüntülenir.
`getch()`	Klavyeden bir karakter okur ve ENTER tuşuna basılmaksızın okunan karakteri bir değişkene aktarır. Klavyeden girilen karakter ekranda görüntülenmez.
`putchar()`	Ekrana bir karakter yazar.
`gets()`	Klavyeden girilen stringi bir değişkene aktarır.
`puts()`	Bir stringi ekrana yazar.

Aşağıdaki program, klavyeden girilen bir stringi ekranda görüntüler.

```
#include <stdio.h>

void main()
{
    char str[80];
    printf("Bir cümle yaziniz:\n");
    gets(str);
    puts(str);
}
```

Program çıktısı:

```
Bir cümle yazınız:
Pamukkale Üniversitesi.
Pamukkale Üniversitesi.
```

Bu programda, `gets` fonksiyonu kullanılarak, klavyeden girilen "`Pamukkale Üniversitesi.`" stringi, `str` değişkenine aktarılmakta, bu string `puts` fonksiyonu ile ekranda görüntülenmektedir. `gets` fonksiyonunun özelliği, ENTER tuşuna basılıncaya kadar girilen sözcüklerin hepsini değişkene aktarmasıdır. `scanf` fonksiyonu kullanılarak ancak tek sözcük girilebilmektedir. `scanf` fonksiyonu ilk boşluktan (bu boşluk dahil) sonraki karakterleri kabul etmez.

Aşağıdaki programda da `gets`, `puts` ve `printf` fonksiyonları birlikte kullanılmaktadır.

```
#include <stdio.h>
#include <conio.h>
void main()
{   char ad[80];
    printf("Adinizi ve soyadinizi giriniz:");
    gets(ad);
    printf("\n") ;
    printf("Merhaba, ");
    puts(ad);
    printf("\n") ;
    printf ("Bir tuşa basınız...");
    getch();
}
```

Program çıktısı:

```
Adınızı ve soyadınızı giriniz: Kadir YALDIR

Merhaba, Kadir YALDIR

Bir tuşa basınız...
```

Bu program çalıştırılıp yukarıdaki görüntüler elde edildikten sonra klavyeden bir tuşa basılmasını bekler ve klavyede herhangi bir tuşa basıldıktan sonra programın çalışması sona erer. Programın sonunda kullanılmış olan `getch()` tuşunun fonksiyonu budur. En son olarak basılan karakter ekranda görüntülenmez.

`getchar()` fonksiyonu, klavyeden girilen tek karakteri okur ve bir değişkene aktarır. Girilen karakterden sonra ENTER tuşuna basılması gereklidir. Aşağıdaki programda `getchar()` fonksiyonu kullanılmıştır.

```
#include <stdio.h>
void main()
{   char c;
    printf("Bir harf giriniz:");
    c = getchar();
    printf("\n");
    putchar(c);
}
```

Program çıktısı:

```
Bir harf giriniz: A

A
```

Bu programda `getchar()` fonksiyonuyla, klavyeden girilen A harfi `c` değişkenine aktarılmış, daha sonra `putchar()` komutu ile bu harf ekranda görüntülenmiştir. `getchar()` fonksiyonu, gerçekte klavyeden girilen ve bilgisayarın ana belleğine (buffer-tampon) alınan birden fazla harfi sırayla okuma özelliğine de sahiptir. Döngüler (loop) konusu ayrıntılı bir şekilde işlenirken, bu konuda örnek program verilecektir.

Aşağıdaki programda, `getche()` ve `getch()` fonksiyonları kullanılarak klavyeden birer harf okunmuştur.

```
/*getch ve getche fonksiyonları */
#include <stdio.h>
#include <conio.h>
void main()
{   char a,b;
    printf("Bir harf tuşuna basınız:");
    a = getche();       /* Basılan tuş ekranda görünür */
    printf("\n") ;
    printf("%c tuşuna bastınız..\n",a);
    printf ("\n") ;
    printf("Bir harf tuşuna daha basınız:");
    b = getch();        /* Basılan tuş ekranda görünmez */
    printf("\n") ;
    printf("%c tuşuna bastınız..\n",b);
}
```

Program çıktısı:

```
Bir harf tuşuna basınız: P

P tuşuna bastınız. .

Bir harf tuşuna daha basınız:

P   tuşuna bastınız..
```

`a=getche()` işlemiyle, `a` değişkenine aktarılan P harfi, tuşa basıldığı anda da ekranda görünürken, `b=getch()` işlemiyle b değişkenine aktarılan Q harfi; tuşa basıldığı anda ekranda görünmemektedir. `printf` fonksiyonu kullanılarak `a` ve `b` değişkenlerindeki karakterler ekranda görüntülenirken, `char` tipi oldukları için format tanımlayıcı olarak "`%c`" kullanılmıştır.

`strlen()` fonksiyonu, bir değişkende tutulan stringin kaç karakterden oluştuğunu bulur. Aşağıdaki programda, `strlen()` fonksiyonu kullanılmıştır.

```
#include <stdio.h>
#include <string.h>
void main()
{   char c[80];
    printf("Bir cümle yazınız :");
    gets(c);
    printf("Cümledeki karakter sayısı: %d\n", strlen(c));
}
```

Program çıktısı:

```
Bir cümle yazınız: Bugün Salı.
Cümledeki karakter sayısı: 11
```

`strlen()` fonksiyonu, `string.h` header dosyasında bulunmaktadır ve bu nedenle, programın başında `string.h` header dosyası programa dahil edilmiştir.

Yukarıdaki programda, `gets(c)` fonksiyonu kullanılarak, klavyeden girilen cümle, `c` değişkenine aktarılır. `strlen(c)` fonksiyonu ile de, bu cümledeki karakterler sayılır. Sözcükler arasındaki boşluk ve cümle sonundaki nokta da bu sayıya dahildir. `strlen()` fonksiyonu ile üretilen sayı integer tipi olduğu için, `printf` fonksiyonu kullanılarak yapılan görüntülemede format tanımlayıcı olarak "`%d`" kullanılmıştır.

C programlama dilinde, klavyeden girilen bütün karakterlerin decimal (ondalıklı) olarak bir karşılıkları bulunmaktadır. Aşağıdaki programda, klavyeden girilen karakterlerin decimal olarak karşılıkları görüntülenmektedir.

```
#include  <stdio.h>
#include <conio.h>
void main()
{   char ch;
    printf ("Lutfen bir harf tusuna basiniz:");
    ch = getche();
    printf("\n") ;
    printf("ASCII olarak  :%c\n", ch);
    printf("DECIMAL olarak:%d\n", ch);
}
```

Program çıktısı:

```
Lutfen bir harf tusuna basiniz: A

ASCII olarak:  A
DECIMAL olarak:65
```