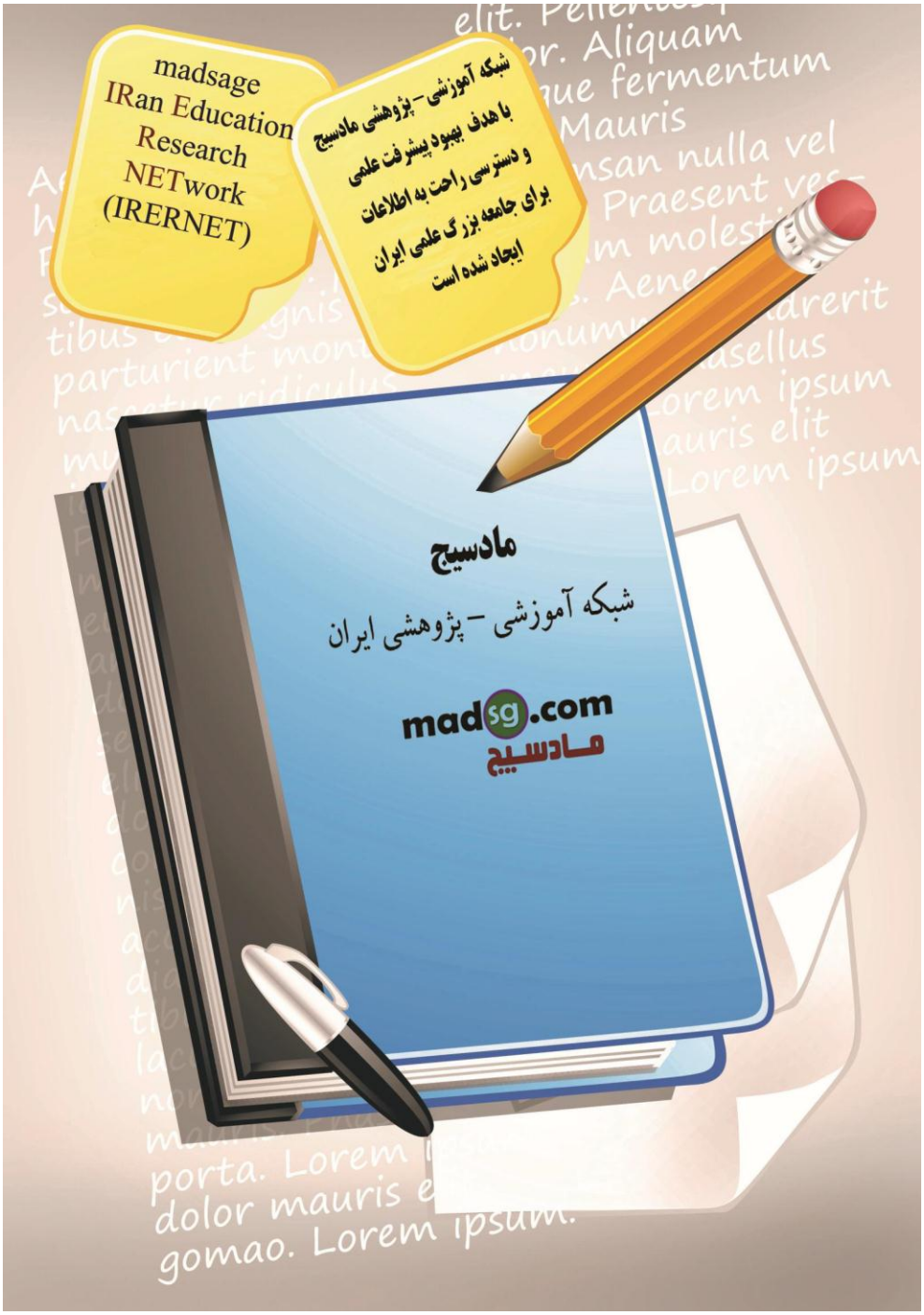
madsage
IRan Education
Research
NETwork
(IRERNET)
شبکه آموزشی-پژوهشی مادسیج
با هدف بهبود پیشرفت علمی
و دسترسی راحت به اطلاعات
برای جامعه بزرگ علمی ایران
ایجاد شده است
مادسیج
شبکه آموزشی -پژوهشی ایران
madsg.com
مادسیج

فصل 3
توابع ورودي و خروجي

هدف کلی
آشنایی با هشت تابع ورودی و خروجی زبان C

هدفهای رفتاری
انتظار می‌رود پس از مطالعة این فصل دانشجو:
1. با کاربرد و ویژگیهای تابع خروجی ()printf آشنا شود.
2. فرامین فرمت را در تابع آرگومان‌دار ()printf بشناسد.
3. با کاربرد و ویژگیهای تابع ورودی ()scanf آشنا شود.
4. تفاوت و تشابه توابع ()printf و ()scanf را بداند.
5. با کاربرد و ویژگیهای تابع ورودی ()getchar آشنا شود.
6. با کاربرد و ویژگیهای تابع خروجی ()putchar آشنا شود.
7. با کاربرد و ویژگیهای تابع ورودی ()getche آشنا شود.
8. با کاربرد و ویژگیهای تابع ورودی ()getch آشنا شود.
9. با کاربرد و ویژگیهای تابع ورودی ـ خروجی ()gets و ()puts آشنا شود.

مقدمه
زبان C با مجموعه‌اي از توابع کتابخانه‌اي همراه است که تعدادي از آنها تابع ورودي و خروجي‌اند. برخي از اين توابع متداول را که اغلب در کتابخانه يا فايل stdio.h قرار دارند در اين فصل بررسي مي‌کنيم.

توابع فرمت‌دار scanf و printf امکان انتقال اطلاعات ميان کامپيوتر و دستگاههاي ورودي و خروجي استاندارد را فراهم مي‌کنند. در واقع اين توابع فرمت‌دار روي داده‌هايي که به شکل پيش‌فرض براي کامپايلر شناخته شده‌اند عمل مي‌کنند. دو تابع getchar وputchar موجب انتقال يك کاراکتر به حافظه و بالعکس مي‌گردند. دو تابع gets و puts نيز ورود و خروج رشته‌ها را ساده‌تر مي‌سازند. برخي از توابع ورودي و خروجي به آرگومان نياز ندارند. بعضي از اين توابع مقداري را برنمي‌گردانند. برخي به صورت مستقل به کار می‌روند و برخي ممکن است در داخل عبارات و دستورها به کار روند.

تابع ()printf
داده‌هاي خروجي استاندارد در کامپيوتر با استفاده از اين تابع نوشته می‌شود. اين تابع به تعداد دلخواه آرگومان می‌پذيرد که آرگومان اول داراي مفهوم خاص است و رشتة فرمت[1] يا رشتة کنترل[2] ناميده مي‌شود. رشتة کنترل در داخل زوج گيومه قرار مي‌گيرد و شامل اطلاعات قالب‌بندي است؛ يعني، مشخص مي‌کند که بايد چند قلم آرگومان داده چاپ شود و فرمت آنها چگونه است. فرم کلي اين تابع به صورت زير است.

printf ("control string", arguments list) ;

يا

printf ("control string", arg1, arg2,..., argn) ;

رشتة کنترل دربردارندة دو نوع اقلام داده است. نوع اول شامل رشته يا کاراکترهايي است که به همان صورت روي صفحه نمايش داده خواهد شد. نوع دوم شامل فرامين فرمت است که با علامت "%" آغاز مي‌گردد و به دنبال آن کد فرمت يا مشخص‌کنندة فرمت مي‌آيد. فرمان فرمت، قالب يا فرمت و تعداد متغيرها يا آرگومانهايي را که بايد چاپ شوند مشخص مي‌سازد و به آن type conversion نيز مي‌گويند و در آن براي مشخص ساختن فرمت هر آرگومان يك گروه از کاراکترها که با "%" آغاز مي‌گردند به کار مي‌رود.

برخي مؤلفان کد فرمت را فرامين فرمت نيز مي‌گويند. فهرست کد يا فرامين فرمت که در فرمان printf به کار می‌روند در جدول 3ـ1 نشان داده شده است.

جدول 3ـ1 فرامين فرمت در تابع ()printf

مفهوم فرمان فرمت	کد تبديل

[1]. format string
[2]. control string

%c	داده به صورت تك‌كاراكتر نمايش داده مي‌شود.
%d	داده به صورت عدد صحيح دهدهي علامت‌دار نمايش داده مي‌شود.
%i	داده به صورت عدد صحيح علامت‌دار نمايش داده مي‌شود.
%f	داده به صورت عدد اعشاري، مميز شناور بدون نما يا توان نمايش داده مي‌شود.
%e	داده به صورت floating-point ولي به فرم نمايي يا علمي نمايش داده مي‌شود.
%g	داده به صورت floating-point به فرم %f يا %e (هركدام كوتاه‌تر باشد) نمايش داده مي‌شود.
%u	داده به صورت عدد صحيح دهدهي بدون علامت نمايش داده مي‌شود.
%s	داده به صورت رشته نمايش داده مي‌شود.
%o	داده به صورت عدد صحيح در مبناي 8 نمايش داده مي‌شود.
%x	داده به صورت عدد صحيح در مبناي 16 و بدون علامت نمايش داده مي‌شود.
L	پيشوندي كه با %o , %x , %u , %d براي معرفي مقدار صحيح بلند يا طولاني به كار مي‌رود (مثل ld).
%p	در مورد داده‌هاي از نوع اشاره‌گر به كار مي‌رود.
%%	علامت % چاپ مي‌كند.

ϖ **مثال1_3** در برنامهٔ زير مقادير متغيرهايي از نوعهاي مختلف در خروجي چاپ مي‌شود.

```
#include<stdio.h>
main ()
{
int x =31 ;
float y =148.5 ;
char z[10] = {"PayamNoor"};
printf(" %d %f %s " , x , y , z) ;
}
```

اولي عدد صحيح دهدهي، دومي از نوع اعشاري و سومي رشته است كه به ترتيب با فرمان فرمت %d , %f , %s مشخص شده‌اند. خروجي دستور مزبور به صورت زير خواهد بود.

31 148.5 PayamNoor

در برنامهٔ مذكور به‌علت وجود يك محل خالي بين فرامين فرمت در رشتهٔ كنترل، بين مقادير چاپ شده نيز فضاي خالي ايجاد شده است.

ϖ

ϖ **مثال 2_3** به‌اين برنامه توجه كنيد.

```
#include<stdio.h>
main ()
{
float x=2.0 , y=3 ;
printf("%f %f %f %f %d", x , y , x+y , x*y, x) ;
}
```

خروجي برنامه به صورت زير خواهد بود.

2.000000 3.000000 5.000000 6.000000 2

در اينجا آرگومانهاي اول و دوم تك‌متغيرند. ولي آرگومانهاي سوم و چهارم عبارات محاسباتي‌اند كه اول مقدار آنها محاسبه مي‌گردد و سپس نتيجه براساس كد فرمت مربوط چاپ مي‌شود. آرگومان آخر به شكل عدد صحيح چاپ مي‌شود.

ϖ

ϖ **مثال 3_3** به برنامهٔ زير توجه كنيد.

```
#include<stdio.h>
main ()
{
double x=50.0 , y=0.25 ;
```

```
printf(" %f %f %f %f\n ", x , y , x*y , x / y) ;
printf(" %e %e %e %e ", x , y , x*y , x / y) ;
}
```

هر دو دستور printf داراي آرگومانهاي يكسان‌اند ولي يكي با فرمان فرمت %f و در ديگري با فرمان فرمت %e چاپ شده‌اند. همچنين علامت "\n" موجب چاپ خروجي در سطر بعدي مي‌گردد.

خروجي اين برنامه به صورت زير خواهد بود.

```
50.000000 0.250000 12.500000 200.000000
5.000000e+01 2.500000e-01 1.250000e+01 2.000000e+02
```

در سطر اول مقادير x , y , x*y , x/y به فرم استاندارد floating point نشان داده شده، درحالي‌كه در سطر دوم همان مقادير به دليل استفاده از فرمان فرمت "%e" به فرم نمايش علمي چاپ شده است.

ملاحظه می‌كنيد كه هر مقدار تا 6 رقم دقت، پس از نقطة اعشار نمايش داده شده است. به هرحال اين تعداد ارقام را مي‌توان با قرار دادن ميزان دقت در رشتة كنترلي تغيير داد.

در فرمان فرمت مي‌توان حداقل پهناي ميدان و تعداد ارقام اعشار، يعني ارقام بعد از مميز، را نيز مشخص كرد. همچنين مي‌توان تعيين كرد كه اطلاعات خروجي از سمت چپ ميدان چاپ (يعني فضاي تعيين‌شده براي چاپ) تراز گردند (در حالت عادي اطلاعات رشته‌اي از طرف چپ و اطلاعات عددي از سمت راست ميدان تراز مي‌شوند). حداقل طول ميدان را مي‌توان با قرار دادن يك عدد صحيح (به عنوان مشخص‌كنندة حداقل فضاي لازم) بين علامت % و كد فرمت مشخص كرد. اين كار موجب مي‌گردد كه اگر طول ميدان بيشتر از طول مورد نياز براي اطلاعات خروجي باشد، فضاي اضافي خالي باقي بماند. ولي اگر طول رشته يا عدد بزرگ‌تر از طول پيش‌بيني‌شده براي آن باشد، طول پيش‌بيني‌شده ناديده گرفته می‌شود و اطلاعات به طور كامل نمايش می‌يابد. اگر بخواهيد در مورد اطلاعات عددي فضاي اضافي با صفر پر شود، سمت چپ عدد m رقم "0" را قرار دهيد. براي مثال %04d موجب مي‌گردد كه اگر مقدار چاپ شده از چهار رقم كمتر باشد، سمت چپ آن به تعداد لازم صفر قرار گيرد به طوري كه مقدار چاپ شده چهاررقمي باشد.

در مورد مقادير floating point براي مشخص ساختن تعداد ارقام بعد از مميز، بايد پس از عدد مشخص‌كنندة طول ميدان، علامت مميز "." و پس از آن نيز يك عدد كه معرف تعداد ارقام اعشار خواهد بود قرار داد. براي مثال كد فرمت %10.4f عدد را با حداقل ده كاراكتر پهناي ميدان و با چهار محل براي ارقام اعشار نمايش خواهد داد.

ϖ

ϖ **مثال 4.3** به برنامة زير دقت كنيد.

```
#include<stdio.h>
main ()
{
int x = 12345 ;
float y =345.125 ;
printf("3d %5d %8d\n\n", x , x , x) ;
printf("%3f %10f %13f\n", y , y , y) ;
printf("%3e %10e %13e\n", y , y , y) ;
printf("%g %10g %13g \n", x , x , x) ;
printf("%3g %10g %13g ", x , x , x) ;
}
```

خروجي اين برنامه به صورت زير خواهد بود.

```
12345 12345 12345
345.125000 345.125000 345.125000
3.45125e+02 3.45125e+02 3.45125e+02
345.125 345.125 345.125
345.125 345.125 345.125
```

در سطر اول، خروجي با استفاده از مينيمم پهناي فيلد (به طول سه كاراكتر، پنج كاراكتر و هشت كاراكتر) چاپ شده است. در هر فيلد تمامي عدد به طور كامل نمايش داده شده است، اگرچه طول ميدان پيش‌بيني شده كمتر از عدد مورد نظر است (مانند فيلد اول كه مينيمم طول پيش‌بيني شده سه كاراكتر است ولي عدد مورد نظر 5 رقمي است).

دومين خروجي در سطر اول با يك محل خالي شروع شده است. اين محل خالي به‌علت وجود يك محل خالي بين فرمان فرمت فيلد اول و دوم در رشتة كنترلي است.

سومين خروجي در سطر اول داراي چهار محل خالي يا blank در سمت چپ است كه يك محل آن به سبب وجود يك محل خالي بين فرمان فرمت فيلد دوم و سوم در رشتة كنترلي است. سه محل

خالي ديگر نيز براي پركردن حداقل فضاي پيش‌بيني شده براي ميدان مورد نظر است (كه در اينجا هشت كاراكتر براي يك عدد پنج رقمي است).

شرايط مشابهي در سطر دوم و سوم خروجي وجود دارد. فقط بايد توجه كرد كه كد فرمت در سطر دوم از نوع f و در سطر سوم از نوع e است.

در سطر چهارم و پنجم همان مقادير با استفاده از كد فرمت g% چاپ شده است. اين فرم نتيجه را كوتاه‌تر نشان مي‌دهد. فواصل بين مقادير چاپ شده نيز براساس درنظر گرفتن حداقل پهناي پيش‌بيني شده در رشتة كنترلي براي آرگومانهاي مورد نظر است.

مي‌توان ماكزيمم تعداد ارقام اعشار در مورد مقادير floating point يا ماكزيمم تعداد كاراكتر براي يك رشته را نيز مشخص كرد. مشخصة مورد نظر براي اين عمل، precision يا دقت ناميده مي‌شود. دقت مورد نظر، يك عدد صحيح بدون علامت است كه قبل از آن نيز يك علامت مميز "." قرار مي‌گيرد. اگر علاوه بر precision حداقل طول ميدان نيز مشخص شده باشد (كه معمولاً نيز همين طور است)، طول ميدان قبل از precision قرار مي‌گيرد و علامت مميز "." بين آن دو درج مي‌شود و كد فرمت پس از اين مجموعه كاراكترها مي‌آيد.

در مورد مقادير floating point اگر دقت پيش‌بيني شده براي ارقام اعشار كوچك‌تر از تعداد ارقام اعشار باشد، جزء اعشار گرد مي‌شود، به طوري كه تعداد ارقام آن با دقت يا precision پيش‌بيني شده مطابقت نمايد.

ϖ

ϖ **مثال 5_3** به‌اين برنامه توجه كنيد.

```
#include<stdio.h>
main ()
{
float x =123.456 ;
printf("%7f %7.3f %7.1f\n", x , x , x) ;
printf("%12e %12.5e %12.3e", x , x , x) ;
}
```

خروجي اين برنامه به صورت زير خواهد بود.

```
123.456000 123.456 123.5
1.234560e+02 1.23456e+02 1.235e+02
```

سطر اول با كد فرمت f ايجاد شده است. در اينجا عدد سوم به دليل وجود مشخصة دقت كه يك رقم اعشار در نظر گرفته شده گرد شده است. همچنين با درنظر گرفتن حداقل طول ميدان (هفت كاراكتر) در سمت چپ عدد سوم، دو محل فضاي خالي به دليل حداقل طول ميدان و يك محل فضاي خالي نيز به دليل وجود يك محل خالي بين فرمان فرمت فيلدهاي دوم و سوم ايجاد شده است.

سطر دوم با كد فرمت نوع e ايجاد شده و داراي مشخصه‌هاي مشابه سطر اول است. در اينجا هم عدد سوم گرد شده است تا با دقت پيش‌بيني‌شده، يعني 3 رقم اعشار، تطابق يابد. همچنين چهار فضاي خالي نيز با در نظر گرفتن حداقل طول ميدان (دوازده كاراكتر) و وجود يك محل خالي بين فرمان فرمت فيلدهاي دوم و سوم در سمت چپ آن ايجاد شده است.

ϖ

ضرورتي ندارد كه مشخصة دقت با حداقل طول ميدان توأم به كار رود، بلكه مي‌توان مشخصة دقت را بدون ذكر حداقل طول ميدان نيز به كار برد. ولي به هرحال نقطة معرف اعشار بايد در جلوي آن به كار رود، مانند دستور زير.

```
printf(" %.4f ", x) ;
```

علاوه بر كاراكترهاي تبديل و پهناي ميدان و شاخص دقت، رشته كنترل يك نشانه يا flag نيز دارد كه در شكل ظاهري خروجي اثر مي‌گذارد. flag بلافاصله بعد از علامت (%) قرار مي‌گيرد. بعضي كامپايلرها اجازه مي‌دهند كه دو flag پشت سر هم درون يک مشخصة تبديل به كار روند. flagهاي متداول در زبان C در جدول 2_3 نمايش داده شده‌اند.

جدول 2_3 Flag هاي متداول

مفهوم و كاربرد	Flag
داده‌ها در داخل ميدان از سمت چپ تراز مي‌شوند. فضاهاي خالي لازم براي پركردن حداقل پهناي ميدان نيز از طرف راست يعني بعد از داده	–

	افزوده مي‌شود.
+	در مورد داده‌هاي عددي، علامت آن (+ يا -) نيز در جلوي آن ظاهر مي‌گردد. بدون استفاده از اين flag فقط در مورد مقادير منفي علامت آنها در خروجي ظاهر مي‌شود.
0	در مورد داده‌هاي عددي موجب مي‌شود كه فضاي خالي سمت چپ به جاي blank با صفر پرشود. البته فقط در مورد داده‌هايي به كار می‌رود كه از سمت راست تراز مي‌شوند.
▪ ▪ blankspace	در مورد هر داده عددي علامت‌دار و مثبت يك blank در جلوي آن ظاهر مي‌شود. اين نشانه اگر با نشانة (+) به كار رود آن را لغو مي‌كند.
# (با نوع تبديل -0 و -x)	باعث مي‌شود كه در جلوي داده‌هاي اكتال 0 و داده‌هاي هگزادسيمال 0x ظاهر شود.
# (با نوع تبديل -g, -f, -e)	باعث مي‌شود كه در تمام اعداد floating-point علامت مميز "." ظاهر شود، حتي اگر عدد مورد نظر جزء اعشار نداشته باشد. همچنين در مورد نوع تبديل g از ناديده گرفته شدن صفرهاي بدون معني سمت راست جلوگيري مي‌كند.

ϖ **مثال 6.3** كاربرد فلاگها در مورد مقادير صحيح و اعشاري در برنامة زير نمايش داده شده است.

```
#include<stdio.h>
main ()
{
int x =123 ;
float y =45.0 , z = -6.7 ;
printf(": %5d %7.0f %10.1e:  \n", x , y , y) ;
printf(": %-5d %-7.0f %-10.1e:  \n", x , y , z) ;
printf(": %+5d %+7.0f %+10.1e:  \n", x , y , z) ;
printf(": %7.0f %#7.0f %7g %#g: " , y , y , z , z) ;
}
```

با اجراي برنامه خروجي زير حاصل مي‌شود كه در آن علامت ":" آغاز ابتداي ميدان و پايان انتهاي ميدان را نمايش مي‌دهد.

```
: 123 45 -6.7+00:
: 123 45 -6.7e+00:
: +123 +45 -6.7e+00:
:+123 +45 -6.7e+00:
: 45 45. -6.7 -6.500000:
```

خط اول خروجي را بدون استفاده از فلاگ نمايش مي‌دهد كه در آن هر عدد درون ميدان مشخص‌شده براي آن، از طرف راست تراز شده است. خط دوم همان اعداد را با استفاده از همان كاراكترهاي تبديل با پيش‌بيني يك فلاگ در هر گروه از رشته فرمت نمايش مي‌دهد، و ملاحظه مي‌كنيد كه‌اين بار به علت وجود فلاگ "-" همة اعداد از سمت چپ تراز شده‌اند. در خط سوم از فلاگ "+" استفاده شده است. در اين حالت اعداد مانند خط اول از سمت راست تراز مي‌شوند. اما علامت مربوط نيز (در هر دو حالت مثبت و منفي) در جلوي آنها ظاهر شده است. در خط چهارم تركيب دو فلاگ "-" و "+" به كار رفته است. در اينجا ضمن اينكه اعداد به دليل وجود فلاگ "-" از سمت چپ تراز شده‌اند، علامت مربوط نيز به‌علت وجود فلاگ "+" در جلوي آنها ظاهر شده است. در خط پنجم دو مقدار مميز شناور را يكبار بدون وجود فلاگ و بار ديگر با استفاده از فلاگ "#" نمايش مي‌دهد و همان طور كه ملاحظه مي‌شود نقش اين فلاگ آن است كه در مورد عدد 45 علامت مميز را نيز منظور داشته است و در مورد عدد دوم صفرهاي بدون ارزش بعد از مميز را نيز در كد فرمت "g" نمايش مي‌دهد.

ϖ

ϖ **مثال 7.3** برنامة زير چاپ اعداد را در مبناي 8 و 10 و 16 نمايش مي‌دهد.

```
#include<stdio.h>
main ()
{
int x = 1234 , y = 0155 , z = 0xa06b ;
printf(": %6u %6o %6x:  \n", x , y , z) ;
```

```
printf(": %-6u %-6o %-6x:  \n", x , y , z) ;
printf(": %#6u %#6o %#6X:  \n" , x , y , z) ;
printf(": %06u %06o %06X:  ", x , y , z) ;
}
```

خروجي برنامه به صورت زير خواهد بود كه در اينجا نيز علامت ":" ابتداي ميدان و پايان ميدان را در هر خط نمايش مي‌دهد.

```
: 1234 155 a06b:
:1234 155 a06b:
: 1234 0155 0XA06B:
:001234 000155 00A06B:
```

خط اول بدون استفاده از فلاگ، اعداد را بدون علامت و به ترتيب در مبناهاي 10، 8 و 16 در خروجي نمايش مي‌دهد. خط دوم همان داده‌ها را با همان كاراكتر تبديل و با استفاده از فلاگ "-" نشان مي‌دهد كه درنتيجه اعداد در فضاي پيش‌بيني‌شده براي آنها از سمت چپ تراز شده‌اند. در خط سوم از فلاگ "#" استفاده شده است. اين فلاگ موجب مي‌گردد كه در جلوي اعداد در مبناي 8 و 16 به‌ترتيب "0" و "0x" ظاهر شود. همچنين به سبب استفاده از حرف بزرگ "X" در كاراكتر تبديل، حروف موجود در اعداد مبناي 16، در خروجي به صورت حروف بزرگ (يعني 0XA06B) ظاهر شده‌اند. خط آخر نقش استفاده از فلاگ "0" را نمايش مي‌دهد. اين فلاگ موجب مي‌گردد كه سمت چپ اعداد به تعداد لازم با صفر پر شود. در اينجا نيز به علت استفاده از حروف بزرگ "X" در كاراكتر تبديل، حروف موجود در اعداد مبناي 16، در خروجي به صورت حروف بزرگ ظاهر شده‌اند.

ϖ

تابع scanf()

در زبان C داده‌هاي ورودي مي‌توانند به كمك تابع كتابخانه‌اي scanf از طريق دستگاه ورودي استاندارد وارد كامپيوتر شوند. تابع scanf نيز تابع فرمت‌دار و مشابه تابع printf است ولي در جهت عكس عمل مي‌كند. به كمك اين تابع مي‌توان داده‌هاي عددي، كاراكترها، رشته‌ها يا تركيبي از آنها را وارد كامپيوتر كرد. فرمت اين تابع مشابه فرمت تابع printf و فرم كلي آن به صورت زير است.

```
scanf ("control string", arguments list) ;
```

يا

```
scanf ("control string", argl , arg2 ,..., arg n) ;
```

در اينجا نقش رشتة كنترل مشابه تابع printf و شامل اطلاعات قالب‌بندي خاص است. مشابه printf اين تابع نيز مي‌تواند هر تعداد آرگومان را دارا باشد، كه در آن اولين آرگومان رشتة فرمت يا رشتة كنترل است. همچنين اين تابع، اغلب همان كد فرمت تابع printf را به كار مي‌برد؛ براي مثال كدهاي فرمت %d , %f , %c , %s كه به ترتيب براي خواندن داده‌هايي از نوع مقادير صحيح، اعشاري، كاراكتر و رشته به كار مي‌روند. تفاوت مهم بين اين دو تابع آن است كه در جلوي آرگومانها، اپراتور آدرس يعني "&" نيز قرار مي‌گيرد.

البته اگر بخواهيد مقداري را براي متغير رشته‌اي بخوانيد، نيازي به اپراتور "&" نخواهد بود زيرا رشته‌ها در زبانِC به صورت آرايه‌اي از نوع كاراكتر معرفي مي‌گردند و نام آرايه نيز معرف آدرس آرايه (يعني آدرس اولين عنصر آن) است.

ϖ **مثال 8_3** برنامة زير نحوة كاربرد عملگر & را در تابع scanf نشان مي‌دهد.

```
#include<stdio.h>
main ()
{
int x ;
char name[6] ;
scanf("%d" , &x) ;
scanf("%s", name) ;
printf("%d %s", x , name) ;
}
```

دستور scanf اول سيستم را هدايت مي‌كند كه داده ورودي را به صورت عدد صحيح از طريق ترمينال دريافت كند و اين مقدار را در متغير x ذخيره كند. دستور scanf دوم به دليل استفاده از آرايه، بدون عملگر & به كار مي‌رود و اگر در اين برنامه براي متغير name رشتة "book" را وارد كرده باشيم، خروجي آن كلمة book خواهد بود.

ϖ

جدول 3_3 فرامين يا كاراكترهاي فرمت براي داده‌هاي ورودي را كه كاراكترهاي تبديل نيز ناميده

مي‌شوند نشان می‌دهد.

جدول 3.3 كاراكترهاي فرمت در تابع scanf ()

كد فرمت	شـــرح
%c	داده ورودي به صورت تك‌كاراكتر تعبير مي‌شود.
%d	داده ورودي به صورت عدد صحيح علامت‌دار (در مبناي 10) تعبير مي‌شود.
%i	داده ورودي به صورت عدد صحيح علامت‌دار تعبير مي‌شود.
%u	داده ورودي به صورت عدد صحيح بدون علامت دهدهي تعبير مي‌شود.
%f , %e, %g	داده ورودي به صورت عدد صحيح اعشاري با ممیز شناور (floating_point) تعبير مي‌شود.
%h	داده ورودي به صورت عدد صحيح كوتاه (short integer) تعبير مي‌شود.
%s	داده به صورت رشته تعبير مي‌شود. ورودي با يك كاراكتر non_white_space آغاز مي‌گردد و با اولين كاراكتر white_space خاتمه مي‌پذيرد (به پايان رشته به طور خودكار كاراكتر "\0" افزوده خواهد شد).
%0	داده ورودي به صورت عدد صحيح در مبناي 8 تعبير مي‌شود.
%x , %X	داده ورودي به صورت عدد صحيح در مبناي 16 تعبير مي‌شود.
%p	داده ورودي اشاره‌گر تعبير مي‌شود.

ϖ **مثال 9.3** برنامة زير يك خط متن حداكثر به طول 79 كاراكتر را مي‌خواند و آن را به همان صورت چاپ مي‌كند.

```
#include<stadio.h>
main () /* read a line of text */
{
char line[80] ;
int count , k ;
/* read in the line */
for (k=0 ; line[k]=getchar ()!='\n' ; + +k)
count = k ;
for (k=0 ; k<count ; + +k)
putchar(line[k]) ;
}
```

در حلقة for، شمارندة k از صفر شروع مي‌شود و مقدار آن در هر تكرار يك واحد افزايش مي‌يابد و در هر تكرار يك كاراكتر با تابع getchar از طريق ورودي استاندارد دريافت مي‌شود و به line[k] نسبت داده مي‌شود و وقتي كه كاراكتر خط جديد (يعني \n) وارد شد، عمل ورود كاراكترهاي رشته خاتمه مي‌يابد كه در اين لحظه مقدار k برابر تعداد كاراكترهاي واقعي رشته خواهد بود. سپس در حلقة بعدي محتواي آراية [ ]line كه دربردارندة رشتة دريافت شده است چاپ مي‌گردد (دو تابع getchar و putchar دوباره بررسي خواهد شد). راه ديگر براي ورود رشته‌ها به حافظة كامپيوتر استفاده از تابع gets است كه در مبحث رشته‌ها بحث می‌كنيم.

براي خواندن رشته‌هايي كه در آنها فضاي خالي (space يا blank) وجود داشته باشد، مي‌توان به طريقي از تابع scanf نيز استفاده كرد. براي اين كار مي‌توان به جاي كاراكتر تبديل نوع s در رشتة كنترلي، دنباله‌اي از كاراكترها را در داخل كروشه به صورت [...] قرار داد كه در اين صورت رشتة مورد نظر هريك از كاراكترهاي موجود در داخل كروشه ازجمله blank را شامل می‌شود.

با چنين روشي وقتي كه برنامه اجرا مي‌گردد، تا زماني كه كاراكترهاي متوالي خوانده شده از طريق دستگاه ورودي با يكي از كاراكترهاي موجود در درون كروشه‌ها يكسان باشد، عمل خواندن رشته‌ها ادامه مي‌يابد. فضاي خالي نيز در داخل رشته‌ها منظور می‌شود. به محض اينكه كاراكتري خوانده شود كه در داخل كروشه‌ها وجود نداشته باشد، عمل خواندن خاتمه مي‌پذيرد. درضمن يك كاراكتر null به طور خودكار به پايان رشته افزوده می‌شود.

ϖ

ϖ **مثال 10.3** برنامة زير كاربرد تابع scanf را براي خواندن رشته‌هايي كه شامل حروف بزرگ و فضاي خالي است نشان مي‌دهد. طول اين رشته با درنظر گرفتن كاراكتر پايان رشته 80 كاراكتر خواهد بود.

```
#include<stdio.h>
main ()
{
char line[80] ;
.....
```

```
scanf("%[ ABCDEFGHIJKLMNOPQRSTUVWXYZ ]", line) ;
.....
}
```

حال اگر از طريق ورودي، رشتة COMPUTER SCIENCE وارد شود، وقتي كه برنامه اجرا مي‌گردد، تمامي رشتة مزبور به آراية line نسبت داده مي‌شود. به هرحال اگر يكي از حروف رشتة مزبور به حرف كوچك تايپ شود، ورود رشته در همان كاراكتر خاتمه مي‌پذيرد. مثلاً اگر در مثال بالا p به‌صورت كوچك تايپ شود، فقط سه حرف com به آراية line نسبت داده مي‌شود و عمل خواندن در حرف چهارم (حرف p) خاتمه خواهد يافت.

راه ديگر آن است كه به جاي اينكه كاراكترهاي مجاز در رشتة مورد نظر را در داخل كروشه ذكر كنيم، فقط كاراكترهايي را كه مجاز نيستيم در رشته‌ها به كار ببريم مشخص می‌كنيم. براي اين كار كافي است كاراكترهاي مورد نظر را به دنبال نماد "^" كه circumflex ناميده مي‌شود، در داخل كروشه قرار دهيم. يعني در اينجا نقش كاراكترهاي كروشه‌اي عكس حالت قبلي است و وجود هركدام از آنها در داخل يك رشته موجب قطع ورود بقية كاراكترهاي رشته مي‌گردد و عمل خواندن رشته خاتمه مي‌پذيرد.

اگر كاراكتر داخل كروشه‌ها كه بعد از "^" مي‌آيد، فقط كاراكتر خط جديد "\n" باشد، رشته‌اي كه از طريق دستگاه ورودي استاندارد وارد مي‌شود هر كاراكتر اسكي به جز كاراكتر خط جديد را شامل می‌شود. بنابراين، كاربر مي‌تواند هرچه خواست به‌عنوان كاراكترهاي رشته وارد كند و در پايان كليد Enter را فشار دهد. اين كليد كاراكتر خط جديد را صادر مي‌كند و درنتيجه پايان رشته را اعلام خواهد كرد.

ϖ

ϖ **مثال 11_3** فرض كنيد كه يك برنامة C شامل دستورهاي زير است.

```
#include<stdio.h>
main ()
{
char line[80] ;
.....
.....
scanf("%[^\n]", line) ;
.....
.....
}
```

وقتي كه تابع scanf در برنامة بالا اجرا مي‌گردد، رشته‌اي به طول نامشخص (ولي حداكثر 79 كاراكتر) از طريق دستگاه ورودي استاندارد وارد مي‌گردد و به line نسبت داده مي‌شود. هيچ گونه محدوديتي در مورد كاراكترهاي تشكيل‌دهندة رشته وجود نخواهد داشت، فقط بايد در يك خط بگنجد.

براي مثال رشتة زير از طريق صفحه‌كليد وارد و به line نسبت داده می‌شود.

WE LEARN MATHEMATICS.

ϖ

تابع getchar()

براي خواندن يك كاراكتر از دستگاه ورودي، مي‌توان علاوه بر تابع scanf از تابع getchar نيز استفاده كرد. تابع مزبور كه جزء كتابخانة I/O زبان استاندارد C است، كاراكتری از دستگاه ورودي استاندارد كه معمولاً صفحه‌كليد است مي‌خواند. اين تابع آرگومان ندارد و به طور متعارف در يك دستور انتساب يا جايگذاري به كار می‌رود و كاراكتر دريافتي از ورودي را به متغيري كه در سمت چپ دستور جايگذاري مورد نظر است اختصاص مي‌دهد. شكل كلي آن به صورت زير است.

```
character variable = getchar() ;
متغير كاراكتري = getchar() ;
```

که در آن متغير كاراكتري نام متغيري از نوع كاراكتر است كه بايد از قبل توصيف شده باشد.

ϖ **مثال 12.3** به دستورهاي زير توجه كنيد.

```
char ch ;
ch = getchar() ;
```

در عبارت اول، متغير ch از نوع كاراكتر توصيف شده است. وقتي كه اجراي برنامه به دستور دوم برسد، برنامه منتظر فشار دادن كليدي از صفحه‌كليد مي‌شود. حال كاراكتر كليد فشار داده شده، به متغير ch اختصاص می‌يابد. چنانچه متغير ch از نوع int معرفي گردد، كد اسكي كاراكتر مربوط به كليد فشار داده شده، به آن متغير اختصاص مي‌يابد.

اگر هنگام خواندن كاراكتر با تابع getchar، شرايط پايان فايل پيش آيد مقدار سمبوليكي EOF به طور خودكار برگشت داده می‌شود (اين مقدار در داخل فايل stdio.h اختصاص می‌يابد. به طور متعارف مقدار -1 به EOF اختصاص داده مي‌شود، اگرچه ممكن است اين مقدار از كامپايلری به كامپايلر ديگر فرق كند). ظاهر شدن EOF به‌اين طريق، راه ساده‌اي براي تشخيص پايان فايل در هنگام اجراي آن است (در اين مورد، در مبحث فايلها بيشتر بحث خواهيم كرد. لذا به هيچ وجه نگران آن نباشيد). مي‌توان تابع getchar را نيز براي خواندن رشتة چند كاراكتري به صورت حلقة تكرار به كار برد كه در هر تكرار يک كاراكتر را بخواند.

ϖ

تابع putchar()

اين تابع براي شمارش يك كاراكتر روي خروجي استاندارد كه معمولاً صفحه نمايش است به كار می‌رود و نقش آن مشابه تابع getchar اما در جهت عكس است. طبيعي است كاراكتري كه انتقال مي‌يابد به صورت ثابت كاراكتر يا متغيری از نوع كاراكتر كه آرگومان تابع مزبور است به كار می‌رود. شكل كلي اين تابع به صورت زير است.

```
putchar (character variable) ;
```

```
putchar (متغير كاراكتري) ;
```

البته مي‌توان ثابت كاراكتري را نيز به عنوان آرگومان تابع مزبور به كار برد. اين تابع با استفاده از آرايه يک رشته را در خروجي چاپ می‌كند.

ϖ **مثال 13.3** برنامة زير به تدريج در هر بار يك كاراكتر مي‌خواند و سپس آن را در خروجي چاپ مي‌كند.

```
#include<stdio.h>
main ()
{
char ch ;
while (1)
{
ch = getchar() ;
putchar(ch) ;
}
}
```

در اين برنامه ch كاراكتر اعلان شده است. هر بار كه يك كاراكتر از طريق دستگاه ورودي استاندارد خوانده مي‌شود، به همان طريق به خروجي انتقال مي‌يابد. اما به لحاظ اينكه عبارت مربوط به while، يعني مقدار داخل پرانتز بعد از while، برابر "1" و هميشه غيرصفر است، براساس قوانين زبان C، اين عبارت هميشه درست يا true است. بنابراين ساختار while(1) حلقه‌ای بي‌نهايت است و تنها راه متوقف ساختن برنامه وقفه‌ای است كه با كليدهاي control-c عملي خواهد شد.

راه ديگر براي نوشتن برنامة بالا به صورت زير است.

```
#include<stdio.h>
main ()
{
int ch ;
while ((ch=getchar()) != EOF)
putchar(ch) ;
}
```

اين برنامه تركيب دو عمل در يك دستور را در حلقه نشان مي‌دهد. گفتيم EOF در برنامة بالا

علامت سمبوليك پايان فايل است. آنچه در واقعيت نشانة پايان فايل را نشان می‌دهد تابع سيستم است. براي اين كار اغلب عدد 1- به كار می‌رود، ولي سيستمهاي مختلف ممكن است مقادير متفاوتي داشته باشند. با گنجانيدن فايل stdio.h و به كار بردن ثابت سمبوليكي EOF، برنامه را قابل حمل يا قابل اجرا ساخته‌ايم. يعني، فايل مبنا روي سيستمهاي مختلف بدون تغيير اجرا می‌شود.

ملاحظه مي‌كنيد كه در روش اخير، متغير ch به جاي char به صورت int معرفي شده است. هرچه براي نشان دادن پايان فايل به كار می‌رود، نمي‌تواند مقداري باشد كه يك كاراكتر را معرفي نمايد. حال چون C به‌صورت int معرفي شده است، مي‌تواند مقادير تمام كاراكترهاي ممكن و همين طور مقدار ويژة EOF را نگهداري كند.

همان طور كه گفتيم، هر دو تابع getchar و putchar در فايل stdio.h تعريف شده‌اند و ممكن است در بعضي سيستمها در فايلهاي ديگري نيز مانند فايل conio.h تعريف شده باشند.

ϖ

ϖ **مثال 14.3** برنامة زير يك خط متن را از ورودي با حروف كوچك دريافت و آن را به حروف بزرگ تبديل مي‌كند.

```
#include<stdio.h>
main ()
{
char line[80] ;
int count , k ;
/* read in the line */
for (k=0 ; (line[k]=getchar())!='\n' ; + + k) ;
count = k ;
/* write out the line in upper-case */
for(k=0 ; k<count ; + +k)
putchar(toupper(line[k])) ;
}
```

در برنامة بالا از حلقة for و تابع كتابخانه‌اي toupper استفاده شده است. نقش اين تابع آن است كه حروف كوچك را به بزرگ تبديل مي‌كند. بنابراين اگر حروف ورودي هنگام تايپ حروف بزرگ يا ارقام و مشابه آن باشند، به شكل اولية خود نمايش داده خواهند شد. براي مثال اگر ورودي به صورت Advanced programming باشد، خروجي به صورت ADVANCED PROGRAMMING خواهد بود.

ϖ

ϖ **مثال 15.3** برنامة زير يك خط از متن را مي‌خواند و در آن هر كاراكتر را (به غير از كاراكتر فضاي خالي يا space) به كاراكتر بعدي تبديل مي‌كند و نمايش مي‌دهد (درواقع متن را به شكلی به صورت رمز در مي‌آورد و نمايش مي‌دهد).

```
#include<stdio.h>
#define space ' '
main ()
{
char ch ;
ch = getchar () ; /* read a character from i/o */
while(ch!='\n') /*while not end of line */
{
if (ch= =space) /* leave the space */
putchar(ch) ; /* character unchanged */
else
putchar(ch+1) ; /* change other characters */
ch = getchar() ; /* get next character */
}
}
```

برای مثال اگر computer science2 ورودي باشد، خروجي dpnqvufs tdjfodf3 خواهد بود.

با تركيب دو دستور خواندن و تست كردن پايان متن در يك عبارت، برنامة مزبور را مي‌توان به صورت ساده و فشرده‌تر زير نوشت.

```
#include<stdio.h>
#define space ' '
main ()
```

```
{
char ch ;
while ((ch=getchar()) != '\n')
{
if (ch = = space) /* leave the space */
putchar(ch) ; /* character unchanged */
else
putchar(ch+1) ; /* change other characters */
}
}
```

که در این برنامه دستور while ((ch=getchar()) != '\n') ترکیب دو عمل در يك دستور را نشان مي‌دهد که روشی متداول در زبان C است. این دو عمل آن است که اول به کمك تابع getchar مقداري به ch نسبت داده مي‌شود و سپس مقدار ch با کاراکتر خط جدید مقایسه می‌شود. وجود پرانتز دور عبارت ch = getchar() آن را اپراند چپ اپراتور =! مي‌سازد. اگر آن را حذف کنیم نتیجة مطلوب به دست نمي‌آید زیرا اپراتور =! نسبت به اپراتور = تقدم بالاتري دارد. بنابراین اگر دستور مزبور را به صورت while (ch = getchar()!= '\n') بنویسیم، اول عبارت getchar()!= '\n' ارزيابي مي‌شود که عبارتی رابطه‌اي است (بنابراین کاراکتر خط جدید برنمي‌گرداند، بلکه يك مقدار برمي‌گرداند) که ارزش آن يك یا صفر (درست یا نادرست یعني true یا false) خواهد بود. سپس این مقدار به ch نسبت داده مي‌شود که هدف مورد نظر ما را از دستور مزبور تأمین نمي‌کند.

ϖ

ϖ **مثال 16.3** برنامة زیر کاراکترها را از طریق ورودي صفحه‌کلید دریافت مي‌کند و آنها را به صفحة نمایش مي‌فرستد. این برنامه با دریافت کاراکتر $ از ورودي خاتمه مي‌پذیرد.

```
#include<stdio.h>
main ()
{
char ch ;
while ((ch=getchar()) != '$')
putchar(ch) ;
}
```

در این برنامه نیز از ترکیب دو دستور در یک دستور در درون حلقة while استفاده شده است.

ϖ

در زبان C علاوه بر تابع getchar دو تابع getch و getche نیز براي خواندن يك کاراکتر از ورودي به کار مي‌رود که در ادامه بررسي می‌کنیم.

تابع getche()

اگر بخواهیم کاراکتري به کمك تابع scanf یا تابع getchar خوانده شود، باید پس از تایپ کاراکتر مورد نظر، کلید Enter را نیز استفاده کنیم. یعني، درواقع دو تابع مزبور تا موقعي که کلید برگشت (که به آن carriage return یا به اختصار CR گویند) فشرده نشود ورودي را در بافر نگه مي‌دارند. پس از زدن کلید برگشت، دادة تایپ شده در اختیار برنامه قرار مي‌گیرد. حسن این روش آن است که اگر کلیدي را اشتباه وارد کرده باشیم، مي‌توانیم آن را با backspace تصحیح کنیم. یعني، قبلي را پاك کنیم و دوباره کاراکتر صحیح مورد نظر را تایپ کنیم. عیب این کار آن است که این عمل در محیط محاوره‌اي امروز وقت‌گیر و دردسرزاست. ازاین رو تابع getche به‌وجود آمد که در آن دیگر نیازي به تحریر کلید برگشت یا CR نیست. اشکال این تابع آن است که اگر کاراکتر اشتباه تحریر شود امکان تصحیح وجود ندارد. همچنین کاراکتر تحریر شده، روي صفحة تصویر نمایش داده مي‌شود که‌این عمل echoing نامیده مي‌شود. در واقع حرف e در آخر نام تابع getche به مفهوم echo (عکس‌العمل) است.

تابع getch()

این تابع همانند تابع getche است با این تفاوت که کاراکتر تحریر شده در صفحة تصویر ظاهر نمي‌گردد. در مورد هریك از این سه تابع وقتي که کنترل اجراي برنامه به این توابع مي‌رسد، برنامه منتظر فشردن کلیدي در صفحه‌کلید مي‌شود. اگر متغیر مورد نظر از نوع کاراکتري باشد، یعني در برنامه با فرمت %c توصیف شده باشد، مقدار کاراکتري کلید فشرده شده به‌این متغیر نسبت داده مي‌شود و درصورتي که‌این متغیر از نوع عددي باشد کد اسکي کاراکتر مربوط به کلید فشرده شده در این متغیر قرار مي‌گیرد.

توابع puts() و gets()

اين دو تابع اين امكان را فراهم مي‌سازند كه بتوان رشته‌هايي از كاراكترها را از طريق كنسول خواند يا در خروجي نوشت (دستگاههاي ورودي و خروجي استاندارد را كنسول نامند كه در مورد ريزكامپيوترها معمولاً صفحه‌كليد ورودي استاندارد و مانيتور خروجي استاندارد را تشكيل مي‌دهند).

تابع gets() يك رشته از كاراكترها را كه از طريق صفحه‌كليد وارد مي‌شود، مي‌خواند و آنها را در آدرسي قرار مي‌دهد كه با آرگومانهاي آن تعيين شده است و اشاره‌گری کاراکتری است. کاراکترهاي رشتهٔ مورد نظر را تايپ مي‌كنيد و در پايان، كليد Enter را می‌زنید. با این عمل به طور خودکار کاراکتر null یا '\0' نیز در پایان رشته قرار می‌گیرد. در اینجا اگر کاراکتری اشتباه تایپ شود، مي‌توان آن را قبل از فشردن كليد Enter با استفاده از كليد backspace تصحيح كرد. در واقع در اینجا نیز کاراکترهاي تايپ شده در بافر مي‌ماند و تا موقعي كه كليد برگشت فشرده نشده است در اختيار برنامه قرار نمي‌گيرد.

تابع puts() آرگومانهاي رشته‌اي خود را به صفحه نمايش مي‌فرستند و سپس قلم نوشتار به خط جديد انتقال مي‌يابد.

ϖ **مثال 17.3** برنامهٔ زير رشته‌اي را از طريق صفحه‌كليد مي‌خواند و در آرايهٔ line قرار مي‌دهد. سپس آن را روي خروجي نمايش مي‌دهد.

```
#include<stdio.h>

main()

{

char line[80] ;

gets(line) ;

puts(line) ;

}
```

فراخواني تابع puts در مقايسه با فراخواني printf داراي overhead كمتري است و درنتيجه سريع‌تر از آن عمل مي‌كند زيرا تابع puts فقط يك رشته از كاراكتر را به خروجي می‌فرستد و نمي‌تواند مشابه printf تبديل فرمت انجام دهد. همچنين نمي‌تواند مقادير عددي را به عنوان خروجي داشته باشد. بنابراين چون puts فضاي كمتري مي‌گيرد و سريع‌تر از printf اجرا مي‌گردد، هنگامي كه در برنامه‌سازي حالت خيلي بهينه مورد نظر باشد، از اين تابع استفاده مي‌شود.

ϖ

خودآزمایی 3

1. برنامه‌اي بنويسيد كه با استفاده از تابع printf و تابع puts رشتهٔ زير را در دو خط جداگانه چاپ كند.

"Payam noor university"

2. برنامه‌اي بنويسيد كه كاراكتري از ورودي بخواند و كاراكتر بعدي آن را در خروجي چاپ كند.
3. برنامه‌اي بنويسيد كه عدد صحيح m1 وعدد اعشاري m2، اطلاعات كاراكتري و آدرس متغير m3 را در خروجي چاپ كند.
4. خروجي برنامهٔ زير چيست ؟

```
# include < stdio. h>
main ()
{
double x ;
x = 2e + 004 ;
printf ("\n x1 = %e" , x) ;
printf ("\n x2 = %E" , x) ;
printf ("\n x3 = %g" , x) ;
}
```

5. خروجي برنامهٔ زير چيست ؟

```
# include < stdio. h>
main ()
{
```

```
float x = 123.456 ;
printf ("%f %.3f %7.2f" , x , x , x) ;
}
```

6. برنامه‌اي بنويسيد كه سه عدد صحيح را از ورودي بخواند و ميانگين آنها را چاپ كند.
7. برنامه‌اي بنويسيدكه دو متغير صحيح را از ورودي بخواند و محتويات آنها را بدون استفاده از متغير كمكي عوض كند و نتيجه را در خروجي نمايش دهد.
8. برنامه‌اي بنويسيد كه سن شما را برحسب روز از ورودي بخواند و مشخص كند كه چند سال، چند ماه، چند هفته و چند روز داريد.


شبکه آموزشی- پژوهشی مادسیج
با هدف بهبود پیشرفت علمی
و دسترسی راحت به اطلاعات
برای جامعه بزرگ علمی ایران
ایجاد شده است
madsg.com
مادسیج
IRan Education & Research NETwork
(IRERNET)