www.mohandesyar.com
www.en2.ir

ساختار برنامهٔ C و ورودی - خروجی

پس از اینکه مقدمات زبان C را در فصل اول آموختید، آمادگی دارید تا در این فصل با ساختار برنامه در C آشنا شده، با یادگیری شیوهٔ ورودی - خروجی داده‌ها، برنامه‌های ساده‌ای بنویسید. اما قبل از پرداختن به امکانات ورودی - خروجی زبان C، ساختار یک برنامه را در این زبان تشریح می‌کنیم. ساختاری که در این فصل بررسی می‌شود، ساختار ساده‌ای است که به تدریج با آموختن امکانات دیگری در C، آن را تکمیل خواهیم کرد.

برنامه‌های زبان C، از مجموعه‌ای از دستورات و تعدادی تابع تشکیل می‌شود (شکل ۱-۲). هر تابع، برای حل بخشی از مسئله نوشته می‌شود و دارای نام است. نامگذاری برای توابع، از قانون نامگذاری برای متغیرها تبعیت می‌کند. در این فصل نمی‌خواهیم راجع‌به توابع بحث کنیم بلکه می‌خواهیم متذکر شویم که بدنه اصلی برنامه، تابعی به نام ()main است. به عبارت دیگر، هیچ برنامه‌ای وجود ندارد که فاقد این تابع باشد. بنابراین، تابع ()main یکی از اجزای مهم برنامه زبان C است.

علاوه بر تابع ()main، توابع دیگری از قبل نوشته شده همراه کامپایلر زبان C ارائه می‌شوند و به وفور در برنامه‌ها مورد استفاده قرار می‌گیرند. در واقع، بسیاری از اعمال در برنامه‌نویسی با C، توسط این توابع از پیش نوشته شده انجام می‌شوند. به عنوان مثال، ورود داده‌ها از صفحه‌کلید و نوشتن آنها در صفحه‌نمایش، در C توسط توابعی انجام می‌شود که از قبل نوشته شدند و همراه کامپایلر وجود دارند. بنابراین قبل از پرداختن به برنامه‌نویسی، باید اطلاع داشته باشیم که این توابع در کجا وجود دارند و برنامه، چگونه به آنها دسترسی پیدا می‌کند. این توابع و سایر اطلاعاتی که کامپایلر برای ترجمه برنامه به آنها نیاز دارد، در تعدادی از فایل‌ها به نام فایل‌های سرآیند[1] قرار دارند.

پسوند این فایل‌ها h. است و معمولاً بر روی فهرستی از دیسک به نام INCLUDE قرار دارند (آنها را در نرم‌افزار C خود مشاهده کنید). بنابراین باید راهی برای اتصال این فایل‌ها به برنامه وجود داشته باشد. برای این کار، باید بدانیم هر تابع مورد استفاده در برنامه، در کدام فایل سرآیند قرار دارد و همان فایل را به برنامه اضافه کنیم. به عنوان مثال، تابع ()scanf که برای ورود داده‌ها از صفحه‌کلید مورد استفاده قرار می‌گیرد، در فایلی به نام stdio.h قرار دارد.

```
#include <فایل سرآیند>
int main()
{
    اعلان متغیرها
    دستورات اجرایی
    return 0;
}
```

شکل ۱-۲ ساختار ساده‌ای از برنامه C.

1. header

www.mohandesyar.com

با توجه به توضیحاتی که ارائه شد، مشخص گردید که بخشی از برنامه C باید کار **اتصال** فایل‌های سرآیند را به برنامه انجام دهد. برای این منظور از دستوری به نام **#include** استفاده می‌شود. این دستور که از دستورات **پیش پردازنده** است، معمولاً قبل از تابع ()main قرار می‌گیرد. به طور کلی، باید به این نکته توجه داشته باشید که، پیش‌پردازنده، مترجمی است که با مشاهده دستوراتی که با # شروع می‌شوند، اجرا می‌شود و آنها را به دستورات زبان C تبدیل می‌کند. توجه داشته باشید که این دستور به ; ختم نمی‌شود. نحوه کاربرد این دستور به صورت زیر است:

#include < نام فایل سرآیند >

به عنوان مثال، برای اضافه کردن فایل stdio.h به برنامه، به صورت زیر عمل می‌شود:

#include <stdio.h>

در مورد دستور include# به نکات زیر توجه کنید:

- بین # و include نباید فاصله‌ای وجود داشته باشد.
- بین نام فایل و علائم > و < نباید فاصله‌ای وجود داشته باشد.
- ذکر علائم > و < ضروری است.

در مورد دستورات پیش پردازنده، در فصل ۲۲ به طور کامل بحث خواهد شد. ولی در اینجا به دلیل ضرورت، مختصری راجع به include# بحث شده است. نکته دیگری که در مورد برنامه C باید بدانید این است که، سیستم عامل که اجراکننده تابع ()main است، می‌خواهد بداند که این تابع با موفقیت به پایان رسیده است یا خیر. برای این منظور بهتر است در پایان اجرای برنامه، مقداری به سیستم عامل برگردانده شود تا سیستم عامل متوجه گردد که برنامه با موفقیت به پایان رسیده است. به همین دلیل، آخرین دستور برنامه C، دستور **return 0;** است. این دستور مقدار صفر را به سیستم عامل برمی‌گرداند و سیستم‌عامل متوجه می‌شود که برنامه با موفقیت اجرا شده است. البته وجود این دستور در بعضی از کامپایلرها ضروری، نیست ولی بعضی دیگر از کامپایلرها آن را ضروری می‌دانند. لذا عادت کنید که این دستور را در انتهای برنامه به کار ببرید.

مقداری که به سیستم عامل برگردانده می‌شود، باید از نوع int باشد. لذا تابع ()main که می‌خواهد مقداری از نوع int را برگرداند، خودش باید از نوع int باشد، به همین دلیل تابع ()main را به صورت زیر در برنامه مورد استفاده قرار می‌دهیم:

int main()

www.en2.ir

توجه به این نکته نیز ضروری است که در بعضی از گونه‌های زبان C، برنامه می‌تواند مقداری را به سیستم عامل برنگرداند. در این صورت باید نوع تابع ()main را void تعریف کرد:

void main()

اگر به‌این صورت عمل کنید،دستور **return 0;** را نباید به‌کار ببرید.

چند نکته راجع به شکل ۱-۲ باید مورد توجه قرار گیرد:

۱. دستورالعملهای برنامه، با { شروع شده به } ختم می‌شوند.

۲. برای اضافه کردن چند فایل سرآیند به برنامه، برای هر کدام از آنها باید یک دستور include# گنجانده شود.

www.mohandesyar.com

۳. متغیرهای موردنیاز برنامه، در ابتدای برنامه و بلافاصله پس از { تعریف می‌شوند.
۴. به تدریج که مطالب جدیدی از C می‌آموزید، ساختار برنامه در C کاملتر می‌شود.

اکنون که با ساختار ساده برنامه در C آشنا شدید، باید دستورالعملهای اجرایی C را بیاموزید تا بتوانید برنامه‌های ساده‌ای بنویسید.

ورودی - خروجی داده‌ها

اصولاً، امکانات ورودی - خروجی هر زبان برنامه‌سازی، نمایانگر میزان قابلیت آن زبان در طراحی فرمهای ورود داده‌ها و اخذ گزارش از برنامه است. در C روشهای گوناگونی برای ورودی - خروجی داده‌ها وجود دارد. بعضی از امکانات ورودی - خروجی این زبان را در این فصل و بعضی دیگر را در فصلهای آینده بررسی می‌کنیم. چون در نظر است آموزش C در این کتاب به صورت گام به گام و خودآموز باشد، سعی می‌کنیم از ذکر مفاهیم پیچیده در فصول اولیه کتاب جلوگیری کنیم.

چاپ اطلاعات با تابع printf()

تابع printf() که در فایل stdio.h قرار دارد، برای چاپ اطلاعات در صفحه‌نمایش به کار می‌رود. اگر این تابع با موفقیت اجرا شود، تعداد کاراکترهایی راکه به خروجی منتقل شده‌اند برمی‌گرداند و در صورت بروز خطا، یک عدد منفی را برمی‌گرداند. نحوه کاربرد این تابع به صورت زیر است:

printf(" < عبارت ۱ > " , < عبارت ۲ >)

در این تابع، <عبارت ۲> اطلاعاتی است که باید به خروجی منتقل شوند و <عبارت ۱> می‌تواند شامل موارد زیر باشد:

۱. اطلاعاتی که باید عیناً در خروجی چاپ شوند.

۲. کاراکترهای تعیین کننده فرمت خروجی. این کاراکترها نوع اطلاعاتی راکه در <عبارت ۲> ذکر شده‌اند و باید به خروجی بروند، مشخص می‌کنند. کاراکترهای فرمت با علامت ٪ شروع می‌شوند. به عنوان مثال، ٪f برای چاپ اعداد اعشاری به کار می‌رود. کاراکترهای فرمت در جدول ۱-۲ آمده‌اند.

۳. کاراکترهای کنترلی. این کاراکترها شکل خروجی اطلاعات را مشخص می‌کنند. اینکه آیا تمام اطلاعات در یک سطر باشند یا در چند سطر چاپ شوند، آیا اطلاعات با فاصله خاصی از یکدیگر چاپ شوند یا خیر، و مواردی از این قبیل، توسط کاراکترهای کنترلی مشخص می‌شود. کاراکترهای کنترلی با \ شروع می‌شوند. به عنوان مثال \n موجب می‌شود تا سطر جاری (سطری که فعلاً در حال نوشتن در آن سطر هستیم) خرد شود و چاپ اطلاعات از سطر جدیدی آغاز گردد. کاراکترهای کنترلی در جدول ۲-۲ آمده‌اند.

توجه داشته باشید که در تابع printf()، <عبارت ۲> می‌تواند وجود نداشته باشد. به عبارت دیگر، تابع printf() به صورت زیر نیز قابل استفاده است:

printf (" <عبارت ۱> ")

www.mohandesyar.com

جدول ۱-۲ کاراکترهای‌فرمت در دستور printf() .

کاراکتر	نوع اطلاعاتی که باید به خروجی برود
%c	یک کاراکتر
%d	اعداد صحیح دهدهی مثبت و منفی
%i	اعداد صحیح دهدهی مثبت و منفی
%e	نمایش علمی عدد همراه با حرف e
%E	نمایش علمی عدد همراه با حرف E
%f	عدد اعشاری ممیز شناور
%g	اعداد اعشاری ممیز شناور
%G	اعداد اعشاری ممیز شناور
%o	اعداد مبنای ۸ مثبت
%s	رشته‌ای از کاراکترها (عبارت رشته‌ای)
%u	اعداد صحیح بدون علامت (مثبت)
%x	اعداد مبنای ۱۶ مثبت با حروف کوچک
%X	اعداد مبنای ۱۶ مثبت با حروف بزرگ
%p	Pointer (اشاره‌گر)
%n	موجب می‌شود تا تعداد کاراکترهایی که تا قبل از این کاراکتر به خروجی منتقل شده‌اند شمارش شده در پارامتر متناظر با آن قرار گیرد
%%	علامت ٪

جدول ۲-۲ کاراکترهای‌کنترلی در دستور printf() .

کاراکتر	عملی که باید انجام شود
\f	موجب انتقال کنترل به صفحه جدید می‌شود
\n	موجب انتقال کنترل به خط جدید می‌شود
\t	انتقال به ۸ محل بعدی صفحه‌نمایش
\"	چاپ کوتیشن (")
\'	چاپ کوتیشن (')
\°	NULL (رشته تهی)
\\	back slash
\v	انتقال کنترل به ۸ سطر بعدی
\N	ثابت‌های مبنای ۸ (N عدد مبنای ۸ است)
\xN	ثابت‌های مبنای ۱۶ (N عدد مبنای ۱۶ است)

www.mohandesyar.com

مثال ۱-۲

برنامه‌ای که چگونگی استفاده از تابع printf() را برای چاپ رشته نشان می‌دهد (به تحلیل مثال توجه کنید).

```
#include <stdio.h>
int main()
{
  printf("C is a language.");
  return 0;
}
```

خروجی

C is a language.

تحلیل مثال ۱-۲

در این برنامه، دستور printf() طوری به کار برده شد که فاقد <عبارت ۲> است. در <عبارت ۱> نه کاراکتر ٪ وجود دارد و نه کاراکتر \. لذا هیچ کاراکتر فرمت و کنترلی در این عبارت وجود ندارد. بنابراین، آنچه که در داخل کوتیشن آمده است، عیناً در خروجی چاپ می‌شود.

مثال ۲-۲

برنامه‌ای که با تابع printf() دو رشته را به خروجی می‌برد (به تحلیل مثال توجه کنید).

```
#include <stdio.h>
int main()
{
  printf("C is a language ");
  printf("and it is my favourite.");
  return 0;
}
```

خروجی

C is a language and it is my favourite.

تحلیل مثال ۲-۲

همان‌طور که در خروجی برنامه می‌بینید، هر دو دستور printf() ، خروجی خود را در یک سطر تولید کرده‌اند. علتش این‌است‌که دستور printf() اول، سطری را که اطلاعات خود را نوشت، رد نمی‌کند و دستور printf() دوم بر روی همان سطر عمل می‌کند.

مثال ۳-۲

برنامه‌ای که با استفاده از کاراکتر کنترلی n\ سطر جاری را رد می‌کند.

```
#include <stdio.h>
int main()
{
  printf("C is a language\n ");
  printf("and it is my favourite.");
  return 0;
}
```

www.mohandesyar.com

خروجی

```
C is a language
and it is my favourite.
```

تحلیل مثال ۳-۲

در این برنامه، وجود \n در انتهای دستور printf() اول، موجب شد تا سطری که اطلاعات در آن نوشته شد رد شود و دستور printf() دوم، اطلاعات را از ابتدای سطر جدید چاپ کند.

مثال ۴-۲

برنامه‌ای که اطلاعات کاراکتری، عددی صحیح و اعشاری و آدرس متغیر x را در خروجی چاپ می‌کند. هدف از این برنامه، آشنایی با کاراکترهای فرمت است (به تحلیل مثال توجه کنید).

```
#include <stdio.h>
int main()
{
  int x = 10;
  float y = 15.5;
  char ch = 'a';
  printf("\n x=%d, y=%f, ch=%c", x, y, ch);
  printf("\n address of x is : %p", &x);
  return 0;
}
```

خروجی

```
x = 10 , y = 15.500000 , ch = a
address of x is : FFF4
```

تحلیل مثال ۴-۲

در این برنامه، در دستور printf() کاراکتر \n سطر جاری را رد می‌کند. کاراکترهای =x عیناً در خروجی چاپ می‌شوند و با فرمت %d مقدار x (۱۰) به خروجی می‌رود. سپس کاما (,) و کاراکترهای =y عیناً در خروجی چاپ می‌شوند و با فرمت %f مقدار y (15.5) در خروجی چاپ می‌شود. سپس کاما (,) و کاراکترهای =ch عیناً در خروجی چاپ می‌شود و با فرمت %c کاراکتر ch (حرف 'a') به خروجی می‌رود. در آخرین دستور printf()، آدرس متغیر x در خروجی چاپ شد که این آدرس ممکن‌است در کامپیوتر شما متفاوت باشد.

مثال ۵-۲

برنامه‌ای که با استفاده از کاراکترهای فرمت و کاراکترهای کنترلی، اطلاعاتی را به خروجی می‌برد.

توضیح:

در این برنامه، برای چاپ علامت ٪ از دو علامت ٪٪ استفاده شد و برای ایجاد فاصله بین x و y از کاراکتر کنترلی \t استفاده گردید. \t موجب شد تا y در محل نهم (ابتدای ۸ محل بعدی) چاپ شود.

```
#include <stdio.h>
int main()
{
  int x = 15, y = 20;
```

www.mohandesyar.com


```
    printf("\nx=%%%d\ty=%%%d", x, y);
    return 0;
}
```

خروجی

```
x = %15   y = %20
```

www.en2.ir

مثال ۶-۲

برنامه‌ای که اعدادی را به صورت نماد علمی نمایش می‌دهد. همانطور که در این برنامه می‌بینید، برای چاپ اعداد با نماد علمی از کاراکترهای فرمت %e، %E و %g استفاده می‌شود.

```
#include <stdio.h>
int main()
{
  double d;
  d = 2e+007;
  printf("\nThe value of d is : %e", d);
  printf("\nThe value of d is : %E", d);
  printf("\nThe value of d is : %g", d);
  return 0;
}
```

خروجی

```
The value of d is : 2.000000e + 07
The value of d is : 2.000000E + 07
The value of d is : 2e + 07
```

مثال ۷-۲

برنامه‌ای برای نمایش چگونگی کاربرد کاراکتر فرمت %n در دستور printf().

توضیح:

کاراکتر فرمت %n مشخص می‌کند که تا قبل از رسیدن به این کاراکتر، چند کاراکتر در خروجی چاپ شده‌اند. این تعداد، در آدرس متغیری که در بخش <عبارت ۲> در دستور printf() آمده است قرار می‌گیرد. در این مثال، تعداد ۱۰ کاراکتر (با احتساب فضای خالی) به خروجی رفته‌اند. به همین دلیل مقدار ۱۰ در count قرار گرفت و چاپ شد.

```
#include <stdio.h>
int main()
{
   int count;
   printf("This is a %n test statement", &count);
   printf("\n count = %d", count);
   return 0;
}
```

خروجی

```
This is a test statement
count = 10
```

www.mohandesyar.com

مثال ۸-۲

برنامه‌ای که با استفاده از کاراکترهای کنترلی، کوتیشن دوتایی را در خروجی چاپ می‌کند و به کمک کاراکتر کنترلی \t فاصله‌ای بین اطلاعات خروجی قرار می‌دهد.

```
#include <stdio.h>
int main()
{
    printf("\n\"each\"\t\"word\"\t\"is\"\n");
    printf("\n\"tabed\"\t\"over\"\t\"once\"");
   return 0;
}
```

خروجی

```
"each"  "word"  "is"
"tabed"  "over"  "once"
```

مشاهده صفحه خروجی برنامه

اگر برنامه‌هایی راکه تاکنون در این فصل نوشته شد، در کامپیوتر تان اجرا کرده باشید، دیدید که پس از تولید خروجی برنامه، کامپیوتر سریعاً به برنامه برمی‌گردد. لذا خروجی برنامه را نمی‌توانید مشاهده کنید. خروجی برنامه در صفحه‌ای غیر از صفحه‌ای که برنامه‌تان را تایپ می‌کنید تولید می‌شود. آن صفحه را **صفحه خروجی** می‌نامیم. اما خوب است که برنامه پس از تولید خروجی، در صفحه خروجی باقی بماند تا پس از مشاهدهٔ خروجی، برای برگشتن به برنامه، کلیدی فشار داده شود. برای این منظور می‌توانید از تابع getch() استفاده کنید (به مثال ۹-۲ مراجعه شود). این تابع منتظر فشردن کلیدی از صفحه‌کلید می‌ماند. الگوی این تابع در فایل conio.h قرار دارد. این تابع کاربرد دیگری نیز دارد که در ادامه بحث خواهد شد. توجه داشته باشید که اگر از تابع getch() استفاده نمی‌کنید، پس از اجرای برنامه می‌توانید با کلید ALT+F5 به صفحه خروجی بروید و پس از مشاهدهٔ اطلاعات، با فشردن کلید Enter، به صفحه برنامه برگردید.

پاک کردن صفحه خروجی

اگر چند برنامه را اجرا کنید و هر برنامه خروجی خاص خودش را تولید کند، صفحه خروجی حاوی اطلاعات متعددی خواهد شد. به طوری که به راحتی نمی‌توانید خروجی برنامه را مورد مطالعه و بررسی قرار دهید. بنابراین، بهتر است در هر بار اجرای برنامه، صفحه خروجی پاک شود. برای این منظور از تابع clrscr() به همین صورت استفاده می‌شود. این تابع در فایل conio.h قرار دارد (به مثال ۹-۲ مراجعه شود).

انتقال مکان‌نما در صفحه خروجی

گاهی ممکن‌است بخواهید مکان‌نما را در صفحه‌خروجی به محل خاصی منتقل کنید و اطلاعات را از آنجا دریافت و یا در آنجا چاپ کنید. به عنوان مثال، ممکن است بخواهید عدد x را از سطر ۵ و ستون ۱۰ بخوانید و در سطر ۶ و ستون ۱۰ چاپ کنید. برای این منظور از تابع gotoxy() استفاده می‌شود. الگوی این تابع در فایل conio.h قرار دارد و به صورت زیر است:

www.en2.ir

www.mohandesyar.com


```
gotoxy(int x , int y) ;
```

x شماره ستون و y شماره سطری است که مکان‌نما به آنجا منتقل می‌شود. به عنوان مثال، دستور gotoxy(40, 10); مکان‌نما را به ستون ۴۰ و سطر ۱۰ منتقل می‌کند. (به مثال ۹-۲ مراجعه شود).

مثال ۹-۲

برنامه‌ای که دو مقدار عددی صحیح را تعریف کرده حاصل جمع آنها را به خروجی می‌برد. برنامه، قبل از تولید خروجی، صفحه خروجی را پاک می‌کند و پس از تولید خروجی در ستون ۲۰ و سطر ۱۰، در این صفحه منتظر می‌ماند تا کلیدی فشار داده شود. پس از مشاهده خروجی این برنامه، باید کلیدی را فشار دهید تا به صفحه برنامه برگردید.

```
#include <conio.h>
#include <stdio.h>
int main()
{
   int m, x = 5, y = 10;
   clrscr();
   m = x + y;
   gotoxy(20, 10);
   printf("x=%d, y=%d, sum=%d", x, y, m);
   getch();
   return 0;
}
```

خروجی

```
x = 5 , y = 10 , sum = 15
```

چاپ اعداد نوع short و long

برای چاپ اطلاعات عددی از نوع short و long، از کاراکترهای خاصی استفاده می‌شود. کاراکتر l (ال) به همراه d برای چاپ مقادیر long و کاراکتر h به همراه d برای چاپ مقادیر short به کار می‌رود. ضمناً، کاراکترهای h و l را می‌توان با کاراکترهای فرمت d، i، o، و u نیز به کار برد.

مثال ۱۰-۲

برنامه‌ای که مقادیر long و short int را در خروجی نمایش می‌دهد.

```
#include <conio.h>
#include <stdio.h>
int main()
{
   short int x = 15;
   long  int m = 35789;
   clrscr();
   printf("\n x=%hd, m=%ld", x, m);
   getch();
   return 0;
}
```