

فهرست

فصل اول:

فصل دوم: ساختار برنامه C و ورودی - خروجی

فصل سوم: حلقه های تکرار و ساختارهای تصمیم

فصل چهارم: توابع و کلاس های حافظه

فصل پنجم: آرایه ها و رشته ها

فصل ششم: اشاره گر ها

فصل هفتم: ساختمان ها

فصل هشتم: فایل ها

فصل نهم: توابع کتابخانه ای

فصل دهم: صف، پشته، لیست پیوندی و درخت

فصل یازدهم: روشهای مرتب سازی و جستجو

فصل دوازدهم: ساختمان کامپیوتر و وقفه ها

فصل سیزدهم: مدل های حافظه و مدیریت صفحه کلید

فصل چهاردهم: رمزگذاری و فشرده سازی متن ها

فصل پانزدهم: توابع کتابخانه ای

فصل شانزدهم: گرافیک

فصل هفدهم: مهندسی نرم افزار به کمک C

فصل هجدهم: طراحی مفسر زبان های برنامه سازی

فصل نوزدهم: توابع کتابخانه ای

فصل بیستم: مدیریت منوها

فصل بیست و یکم: ارتباط زبان C با اسمبلی

فصل اول:

۱- مقدمات زبان C

زبان C در سال ۱۹۷۲ توسط دنیس ریچی طراحی شد. این زبان تکامل یافته زبان BCPL می باشد که طراح آن مارتین ریچاردز است. زبان BCPL از زبان B که طراح آن کن تامپسون می باشد، نتیجه شده است. علت نامگذاری C این است که بعد از B طراحی شد.

کسانی که تا حدودی با زبانهای برنامه سازی آشنایی دارند، می دانند که زبان دیگری به نام زبان ++C وجود دارد و آن از C ناشی شده است. ++C علاوه بر ویژگیهای C، ویژگیهای جدیدی دارد که در C موجود نیست.

بعضی از ویژگیهای زبان C عبارتند از:

- زبان C یک زبان میانی است. زبانهای برنامه سازی را می توان به سه دسته تقسیم کرد: زبانهای سطح بالا، زبانهای میانی، زبانهای سطح پایین (جدول ۱-۱). علت میانی بودن زبان C این است که، از طرفی همانند زبان سطح پایینی مثل اسمبلی قادر است مستقیماً به حافظه دستیابی داشته باشد و با مفاهیم بیت، بایت و آدرس کار کند. و از طرف دیگر، برنامه های این زبان، همچون زبانهای سطح بالایی مثل پاسکال، از قابلیت خوانایی بالایی برخوردارند. به عبارت دیگر، دستورالعملهای این زبان، به زبان محاوره ای انسان نزدیک است، که این ویژگی، مربوط به زبانهای سطح بالا است.
- زبان C، یک زبان ساخت یافته است. در این زبان با استفاده از حلقه های تکراری مثل `for`، `while` و `do while` می توان برنامه هایی نوشت که قابلیت خوانایی و درک آنها بالا باشد. بعضی از زبانهای ساخت یافته در جدول ۱-۲ آمده اند.

جدول ۱-۱ سطوح زبانهای برنامه سازی

زبانهای سطح بالا	زبانهای میانی	زبانهای سطح پایین
پاسکال	جاوا	ماکرواسمبلر
ادا	فورث	اسمبلر
ماجولا - ۲	C, (C++)	
کوبول		
بیسیک		

جدول ۱-۲: زبانها از نظر ساخت یافتگی

زبانهای ساخت یافته	زبانهای غیر ساخت یافته
پاسکال	فرترن
ادا	بیسیک
C, (C++)	کوبول
ماجولا - ۲	
جاوا	

جدول ۱-۳: کلمات کلیدی زبان C

Auto	Double	Int	Struct
Break	Else	Long	Switch
Case	Enum	Register	Typedef
Char	Extern	Return	Union
Const	Float	Short	Unsigned
Continue	For	Signed	Void
Default	Goto	Sizeof	Volatile
do	if	static	while

- زبان C، قابل انعطاف و بسیار قدرتمند است. در این زبان، هیچ محدودیتی برای برنامه نویسی وجود ندارد. هر آنچه را که فکر می کنید، می توانید در این زبان پیاده سازی کنید.
- C، زبان برنامه نویسی سیستم است. برنامه های سیستم، برنامه هایی هستند که امکان بهره برداری از سخت افزار و سایر نرم افزارها را فراهم می کنند. بعضی از برنامه های سیستم عبارتند از: سیستم عامل، مفسر^۱، کامپایلر، ویراستارها، واژه پردازها، مدیریت بانکهای اطلاعاتی و اسمبلر.
- ارتباط تنگاتنگی بین زبان C و اسمبلی وجود دارد و به این ترتیب می توان از تمام قابلیت های اسمبلی در زبان C استفاده کرد. چگونگی برقراری ارتباط بین این دو زبان، در فصل ۲۱ به طور مفصل مورد بحث قرار می گیرد.
- C، زبان قابل حمل است. معنای قابلیت حمل این است که برنامه هایی که به زبان C، در یک نوع کامپیوتر (مثل آی. بی. ام) نوشته شدند، بدون انجام تغییرات یا انجام تغییرات اندک، در کامپیوترهای دیگر (مثل VAX و DEC) قابل استفاده اند.
- C، زبان کوچکی است. تعداد کلمات کلیدی^۲ این زبان انگشت شمار است (۳۰ کلمه کلیدی - جدول ۱-۳). تصور نشود که هر چه تعداد کلمات کلیدی زبان بیشتر باشد، آن زبان قدرتمند است. به عنوان مثال، زبان بیسیک در حدود ۱۵۰ کلمه کلیدی دارد و لی قدرت زبان C به مراتب بیشتر از زبان بیسیک است. توجه داشته باشید که بعضی از کامپایلرهای C، علاوه بر این ۳۲ کلمه کلیدی، کلمات دیگری را به زبان اضافه کرده اند (جدول ۱-۴)
- C نسبت به حروف حساس است^۳. یعنی در این زبان، بین حروف کوچک و بزرگ تفاوت است و تمام کلمات کلیدی این زبان با حروف کوچک نوشته می شوند. به عنوان مثال، while یک کلمه کلیدی است ولی WHILE اینطور نیست. توصیه می شود که تمام برنامه های C با حروف کوچک نوشته شوند.
- دستورالعملهای برنامه C دارای ویژگیهای زیر هستند:
 - ۱- هر دستور زبان C به ; ختم می شود.
 - ۲- حداکثر طول یک دستور، ۲۵۵ کاراکتر است.
 - ۳- هر دستور میتواند در یک یا چند سطر ادامه داشته باشد.
 - ۴- در هر سطر می توان چند دستور را تایپ کرد (این کار، توصیه نمی شود).
 - ۵- توضیحات می توانند در بین /* و */ قرار گیرند و یا بعد از // ظاهر شوند:

/* This is a sample comment */

// This is another sample comment

¹ - interpreter

² - key words

³ - case sensitive

۱-۱- انواع داده ها

هدف از برنامه نویسی، ورود داده ها به کامپیوتر، پردازش داده ها و استخراج نتایج است. لذا، داده ها نقش مهمی را در برنامه نویسی ایفا می کنند. یکی از جنبه های زبانهای برنامه سازی که باید دقیقاً مورد بررسی قرار گیرد، انواع داده هایی است که آن زبان با آنها سر و کار دارد. در زبان C، پنج نوع داده وجود دارند که عبارتند از: char، float، double و void. نوع char برای ذخیره داده های کاراکتری مثل 'a'، 'b'، 'x' به کار می رود. نوع int برای ذخیره اعداد صحیح مثل 125، 430، 1650 به کار می رود. نوع float برای ذخیره اعداد اعشاری مثل 15.5، 175.5 و 1250.25 به کار می رود و نوع double برای ذخیره اعداد اعشاری که بزرگتر از float باشند مورد استفاده واقع می شود. نوع void را در جای مناسبی تشریح خواهیم کرد. هر یک از انواع داده های char، int، float و double مقادیری را می پذیرند که ممکن است از پردازنده ای (CPU) به پردازنده دیگر متفاوت باشد. به عنوان مثال، طول نوع int در محیطهای ۱۶ بیتی مثل DOS یا ویندوز ۳/۱، شانزده بیت و در محیطهای ۳۲ بیتی مثل ویندوز NT، سی و دو بیت است. بنابراین، اگر برنامه هایی می نویسید که باید در محیطهای مختلف اجرا شوند، سعی کنید از کوچکترین مقدار انواع در C استفاده نمایید.

۱-۲- متغیرها

متغیر نامی برای کلمات حافظه است که داده ها در آنها قرار می گیرند و ممکن است در طول اجرای برنامه تغییر کنند. برای مراجعه به متغیرها از نامشان استفاده می شود. لذا متغیرها امکان نامگذاری برای کلمات حافظه را فراهم می کنند. برای نامگذاری متغیرها می توان از ترکیبی از حروف a تا z یا A تا Z، ارقام و خط ربط (_) استفاده کرد، به طوری که اولین کاراکتر آنها رقم نباشد.

۱-۲-۱- تعریف متغیرها

همانطور که گفته شد، متغیرها محل ذخیره داده ها هستند و چون داده ها دارای نوع اند، متغیرها نیز باید دارای نوع باشند. به عبارت دیگر، متغیرهای فاقد نوع، در C شناخته شده نیستند. قبل از به کار گرفتن متغیرها، باید نوع آنها را مشخص کرد. نوع متغیر، مقادیری را که متغیر می تواند بپذیرد و اعمالی را که می توانند بر روی آن مقادیر انجام شوند، مشخص می کند. تعیین نوع متغیر را تعریف متغیر گویند. برای تعیین نوع متغیر، به صورت زیر عمل می شود:

نام متغیر نوع داده

برای تعیین نوع بیش از یک متغیر، باید آنها را با کاما از هم جدا کرد.

مثال ۱-۱

تعریف متغیرهای x و y از نوع int، متغیرهای m و n از نوع float، متغیرهای ch1 و ch2 از نوع char، متغیر d1 از نوع double و متغیر p1 از نوع long int.

Int	x,y;
Float	m, n;
Char	ch1,ch2;
Double	d1;
Long int	p1;

۱-۲-۲- مقدار دادن به متغیرها

برای مقدار دادن به متغیرها به سه روش می توان عمل کرد:

- ۱- هنگام تعریف (تعیین نوع) متغیر
- ۲- پس از تعریف نوع متغیر و با دستور انتساب (=)
- ۳- دستورات ورودی

مثال ۲-۱.

مقدار دادن به متغیرها در هنگام تعریف آنها.

```
Int      x, y=5;
Char     ch1='a', ch2='m';
```

دستور اول، دو متغیر X و Y را از نوع int تعریف می کند و مقدار متغیر Y را برابر با ۵ قرار میدهد. دستور دوم متغیرهای ch1 و ch2 را از نوع char تعریف می کند، مقدار ch1 را برابر با 'a' و مقدار ch2 را برابر با 'm' تعیین می کند. کاراکترها در داخل کوتیشن یکسانی (‘) قرار می گیرند.

مثال ۳-۱.

مقدار دادن به متغیرها با دستور انتساب.

```
Int      x,y,m;
Float    f1, f2;
Char     ch1, ch2;
F1= 15.5;
F2= 20.25;
X= y= m = 0;
Ch1= ch2= 'a';
```

سه دستور اول، متغیرها را تعریف می کنند. دستور چهارم، مقدار 15.5 را در f1 و دستور پنجم مقدار 20.25 را در متغیر f2 قرار میدهد. دستور ششم سه متغیر X و Y و m را برابر با صفر قرار میدهد (انتساب چندگانه در C امکان پذیر است). دستور هفتم، حرف 'a' را در متغیرهای ch1 و ch2 قرار می دهد.

مثال ۴-۱. مقدار دادن به متغیرها با دستورات ورودی.

```
Int      x, y;
Scant("%d%d", &x, &y);
```

توجه داشته باشید که در اینجا، متغیرهای X و Y از نوع int تعریف شدند و دستور scanf مقادیر آنها را از طریق صفحه کلید دریافت می کند. فعلاً به چگونگی عملکرد آن کاری نداشته باشید، بلکه فقط این مطلب را یاد بگیرید که، متغیرها با دستورات ورودی، مقدار می گیرند.

۳-۱. تعریف ثوابت

ثوابت مقادیری هستند که در برنامه وجود دارند ولی قابل تغییر نیستند. برای تعریف ثوابت به دو روش عمل می شود: ۱- استفاده از دستور #define و ۲- استفاده از دستور const. برای تعریف ثوابت از طریق دستور #define به صورت زیر عمل می شود:

```
#define <مقدار> <نام ثابت>
```

نامگذاری برای ثوابت از قانون نامگذاری برای متغیرها تبعیت می کند. مقداری که برای ثابت تعیین می شود، نوع ثابت را نیز مشخص می کند. دقت داشته باشید که در انتهاب دستور #define علامت ; قرار نمی گیرد. علتش این است که این دستور، از دستورات پیش پردازنده^۱ است، نه دستور زبان C، پیش پردازنده یک برنامه سیستم است که قبل از ترجمه برنامه توسط کامپایلر، تغییراتی در آن ایجاد می کند. پیش پردازنده مقدار ثابت را که در دستور #define آمده است، به جای نام ثابت در برنامه قرار میدهد و این دستور در زمان اجرا وجود ندارد. نام دیگر ثوابتی که به این صورت تعریف می شوند، ماکرو^۲ است. برای تفکیک اینگونه ثوابت از متغیرهای برنامه، بهتر است نام آنها با حروف بزرگ انتخاب شود.

¹ - preprocessor

² - macro

مثال ۵-۱. تعریف ثوابت PI و M با دستور #define. دستور اول، مقدار M را برابر با ۱۰۰ و دستور دوم مقدار PI را برابر با ۳/۱۴ تعیین می کند.

```
#define M 100
#define pi 3.14
```

برای تعریف ثوابت با دستور const به صورت زیر عمل می شود:

Const <مقدار> = <نام ثابت> <نوع داده>

نام ثابت مثل نام متغیرها انتخاب می شود که مقدار ثابت با علامت = در آن قرار می گیرد.

۴-۱. عملگرها

عملگرها^۱ نمادهایی هستند که اعمال خاصی را انجام می دهند. به عنوان مثال، نماد '+' عملگری است که دو مقدار را با هم جمع می کند (عمل جمع را انجام میدهد). پس از تعریف متغیرها و مقدار دادن به آنها باید بتوان عملیاتی را روی آنها انجام داد. برای انجام این عملیات باید از عملگرها استفاده کرد. عملگرها در زبان C به چند دسته تقسیم می شوند: ۱- عملگرهای محاسباتی. ۲- عملگرهای رابطه ای. ۳- عملگرهای منطقی. ۴- عملگرهای بیتی. عملگرها بر روی یک یا دو مقدار عمل می کنند. مقادیری را که عملگرها بر روی آنها عمل می کنند، عملوند^۲ گویند. به عنوان مثال، در $a+5$ ، متغیر a و مقدار 5 را عملوندهای عملگر + گویند.

۱-۴-۱. عملگرهای محاسباتی

عملگرهای محاسباتی، عملگرهایی هستند که اعمال محاسباتی را روی عملوندها انجام می دهند. این عملگرها در جدول زیر مشاهده می شوند. هر یک از عملگرهای +، -، *، / تقریباً در همه زبانها وجود دارد.

دستورات زیر را در نظر بگیرید:

```
Int x=10, m=10
```

```
X ++;
```

```
++ m;
```

متغیرهای X و m با مقدار اولیه ۱۰ تعریف می شوند. دستور ++ X یک واحد به X اضافه می کند و نتیجه را در X قرار می دهد و دستور ++ m یک واحد به m اضافه کرده نتیجه را در m قرار می دهد. بنابراین در این نوع دستورات، عملگر افزایش (یا کاهش) چه قبل از عملوند باشند و چه بعد از آن، عملکرد آنها یکسان است. اما عملکرد آنها در عبارات محاسباتی با هم متفاوت است.

جدول: عملگرهای محاسباتی

مثال	نام	عملگر
-x یا x-y	تفریق و منهای یکانی	-
X+y	جمع	+
X*y	ضرب	*
x/y	تقسیم	/
x % y	باقیمانده تقسیم	%
-x یا x—	کاهش (decrement)	--
++ x یا X++	افزایش (increment)	++

¹ - Operators

² - operand

۲-۴-۱- تقدیم عملگرها

وقتی در عبارتی، چندین عملگر وجود داشته باشند، ترتیب اجرای عملگرها چگونه خواهد بود؟ برای پاسخ به این سوال، باید تقدم عملگرها را مشخص کرد. برای پی بردن به این مسئله، دستورات زیر را در نظر بگیرید:

Int m, x=6, y=10;

M= x+y/2*3;

به نظر شما، حاصل این عبارت چه خواهد بود؟ اگر ابتدا X با Y جمع شود و حاصل آن بر ۲ تقسیم شود و نتیجه آن در ۳ ضرب شود، حاصل عبارت ۲۴ خواهد بود. اگر ابتدا Y بر ۲ تقسیم شود و حاصل آن با X جمع شود و نتیجه آن در ۳ ضرب شود، حاصل آن ۳۳ خواهد بود. اگر ابتدا Y بر ۲ تقسیم شود و نتیجه آن در ۳ ضرب شود، حاصل آن با X جمع شود، حاصل عبارت، ۲۱ خواهد بود. کدام مقدار درست است؟ برای یافتن پاسخ درست، باید تقدم عملگرها را بدانیم. تقدم عملگرهای محاسباتی در جدول آمده است. همانطور که در این جدول مشاهده می کنید، بالاترین تقدم مربوط به عملگرهای افزایش و کاهش و کمترین تقدم مربوط به عملگرهای + و - است. عملگرهایی که تقدم آنها یکسان است، مثل +، -، یا /، * و / نسبت به هم تقدم مکانی دارند. یعنی، هر کدام که زودتر ظاهر شد، همان عملگر زودتر انجام می شود.

با توجه به مطالب گفته شده، در عبارت $m=x+y/2*3$ ابتدا عملگر / و سپس عملگر * و در انتها عملگر + انجام می شود. به این ترتیب حاصل این عبارت برابر با ۲۱ است.

۳-۴-۱. عملگرهای رابطه ای

عملگرهای رابطه ای، ارتباط بین عملوندها را مشخص می کنند. اعمالی مثل تساوی دو مقدار، کوچکتر یا بزرگتر بودن، مقایسه با صفر، و غیره، توسط عملگرهای رابطه ای مشخص می شود. عملگرهای رابطه ای در جدول زیر آمده اند. د رمورد عملگرهای رابطه ای، شاید با عملگر == آشنایی نداشته باشید. این عملگر در دستورات شرطی برای مقایسه دو مقدار مورد استفاده قرار می گیرد. به عنوان مثال، دستور مقایسه دو مقدار X و Y، باید به صورت آیا $x==y$ است، نوشته شود.

جدول: تقدم عملگرهای محاسباتی

بالاترین تقدم	++--
(منهای یکانی)	-
	*/%
پایین ترین تقدم	+ -

جدول: عملگرهای رابطه ای

عملگر	نام	مثال
>	بزرگتر	$x>y$
>=	بزرگتر یا مساوی	$x>=y$
<	کوچکتر	$x<y$
<=	کوچکتر یا مساوی	$x<=y$
==	متساوی	$x==y$
!=	نامساوی	$x!=y$

جدول: عملگرهای منطقی به ترتیب تقدم

مثال	نام	عملگر
$\neg X$	نقیض (not)	!
$x > y \ \&\& \ m < p$	و (and)	&&
$x > y \ \ m < p$	یا (or)	

۴-۱-۴. عملگرهای منطقی

عملگرهای منطقی بر روی عبارات منطقی عمل می کنند. عبارات منطقی دارای دو ارزش درستی و نادرستی اند. در زبان C، ارزش نادرستی با مقدار صفر و ارزش درستی با مقادیر غیر صفر مشخص می شود. عملگرهای منطقی در جدول زیر آمده است. ترتیب قرار گرفتن آنها در این جدول از تقدم بالا به پایین است.

نتیجه عملگر ! وقتی درست است که عملوند آن دارای ارزش نادرستی باشد. نتیجه عملگر && وقتی درست است که هر دو عملوند ارزش درستی داشته باشند و نتیجه عملگر || وقتی نادرست است که هر دو عملوند ارزش نادرستی داشته باشند (در بقیه موارد نتیجه آن ارزش درستی دارد). به عنوان مثال، دستورات زیر را در نظر بگیرید:

Int x,y, m, p, q;

X=0

Y=1

M= x && y;

P= x || y;

Q= ! x ;

با اجرای دستور چهارم، ارزش دارای m برابر با نادرستی خواهد بود، زیرا X دارای ارزش نادرستی و Y دارای ارزش درستی است و حاصل بر && بر روی آنها، نادرستی است. با اجرای دستور پنجم، P ارزش درستی خواهد شد، زیرا قرار X دارای ارزش نادرستی است و Y دارای ارزش درستی است و عملگر || روی آنها عمل می کند و ارزش درستی را برمی گرداند. چون X ارزش نادرستی دارد، !X دارای ارزش درستی خواهد بود که نتیجه آن در Q قرار می گیرد. چون اغلب، عملگرهای منطقی و رابطه ای با هم مورد استفاده قرار می گیرند درک تقدم آنها از اهمیت ویژه ای برخوردار است، تقدم آنها را در جدول زیر مشاهده می کنید.

جدول: تقدم عملگرهای منطقی و رابطه ای

!	بالاترین
>>=<<=	
= = ! =	
&&	
	پایین ترین

۶-۱-۴. عملگرهای بیتی

وجود عملگرهای بیتی در C موجب شد تا بسیاری از کارهای زبان اسمبلی در C انجام شود. عملگرهای بیتی برای تست کردن، مقدار دادن یا شیفت دادن و سایر اعمال بر روی مقادیری که در یک بایت (char) یا کلمه (int) ذخیره شده اند به کار می روند. عملگرهای بیتی را نمی توان با انواع float، double، long double و void یا سایر انواع پیچیده به کار برد. عملگرهای بیتی در جدول زیر آمده اند.

جدول: عملگرهای بیتی

نام	عملگر
(AND)	&
(OR)	
(XOR)	^
(Not)	~
(right shift)	>>
(left shift)	<<

۷-۴-۱. عملگرهای & و *

همانطور که گفته شد، متغیرها نامی برای کلمات حافظه اند و کلمات حافظه نیز دارای شماره ردیف می باشند که ما آنها را آدرس می نامیم. با استفاده از عملگر & می توانیم به آدرس متغیرها دسترسی داشته باشیم. عملگر * نیز برای دسترسی غیرمستقیم به حافظه مورد استفاده قرار می گیرد. به عنوان مثال، اگر فرض کنیم آدرس متغیر X در p قرار داشته باشد، از طریق آدرس X و با استفاده از عملگر * می توانیم به محتویات X دسترسی پیدا کنیم. با فرض اینکه p می تواند حاوی آدرس X باشد، دستورات زیر را در نظر بگیرید:

$P = \& x;$ آدرس x در p قرار می گیرد
 $* P = 5;$ جایی که آدرس آن در p است (همان X) برابر با ۵ می شود.
 $* M = *p;$ محتویات جایی که آدرسش در p است (۵) در m قرار می گیرد.

۸-۴-۱. عملگر ؟

این عملگر، عبارتی را ارزیابی کرده، بر اساس ارزش آن عبارت (درستی یا نادرستی)، نتیجه عبارت دیگر را در متغیری قرار می دهد:

<عبارت ۳>: <عبارت ۲> ؟ <عبارت ۱> = متغیر

اگر <عبارت ۱> دارای ارزش درستی باشد، مقدار ارزیابی شده <عبارت ۲> در متغیر قرار می گیرد و گرنه مقدار ارزیابی شده <عبارت ۳> در متغیر قرار خواهد گرفت. دستورات زیر را در نظر بگیرید:

Int x, y;
 $X = 5;$
 $Y = x > 5 ? x * 2 : x * 5;$

در این دستورات، موارد زیر را داریم:

<عبارت ۱>: $x > 5$

<عبارت ۲>: $x * 2$

<عبارت ۳>: $x * 5$

<عبارت ۱> دارای ارزش نادرستی است، زیرا X از ۵ بزرگتر نیست. بنابراین مقدار <عبارت ۳> که برابر با $5 \times 5 = 25$ است، در متغیر Y قرار می گیرد.

۹-۴-۱. عملگر کاما (،)

این عملگر برای انجام چند عمل در یک دستور به کار می رود و روش کاربرد آن به صورت زیر است:

<عبارت ۲> و <عبارت ۱> = متغیر

<عبارت ۱> به نحوی با <عبارت ۲> در ارتباط است. به طوری که، ابتدا <عبارت ۱> ارزیابی می شود و <عبارت ۲> می تواند از نتیجه آن استفاده کند و حاصل <عبارت ۲> در متغیر قرار می گیرد. دستورات زیر را در نظر بگیرید:

```
Int x, y;
Y=(x=2, x* 4/2);
```

در این دستورات موارد ذیل را داریم:

<عبارت ۱> : $x = 2$
 <عبارت ۲> : $x * 4/2$

در <عبارت ۱> مقدار ۲ در X قرار می گیرد و این مقدار در محاسبه عبارت $x * 4/2$ شرکت کرده، حاصل آن، یعنی ۴ در Y قرار می گیرد.

۱۰-۴-۱. عملگر sizeof

این عملگر، یک عملگر زمان ترجمه است و می تواند طول یک متغیر یا نوع داده را بر حسب بایت تعیین کند. اگر با کامپوتری کار می کنید و نمی دانید انواع آن، مثلاً نوع int چند بایتی است، با این عملگر می توانید به آن پی ببرید. این عملگر به صورتهای زیر به کار می رود:

```
Sizeof متغیر;
Sizeof (نوع داده);
```

همانطور که در نحوه کاربرد این عملگر می بینید، وقتی برای تعیین طول نوع داده به کار می رود، نوع داده باید در داخل پرانتز قرار گیرد. دستورات زیر را در نظر بگیرید:

```
Int x, y, m;
X = sizeof y;
M = sizeof (float);
```

دستور اول، سه متغیر X و Y و m را از نوع صحیح تعریف می کند. دستور دوم طول متغیر Y را محاسبه کرده و در X قرار می دهد و دستور سوم، طول نوع float را محاسبه کرده و در m قرار می دهد. بعداً که روش چاپ متغیرها را یاد گرفتید، می توانید با چاپ متغیرهای X و m، به طور متغیر Y و نوع float پی ببرید.

۱۱-۴-۱. عملگر ()

پرانتزها عملگرهایی هستند که تقدم عملگرهای داخل خود را بالا می برند. به عنوان مثال، عبارت زیر را در نظر بگیرید:

$$Y = 4 * 2 / (3 + 1) + (6 + (7 - 2))$$

برای ارزیابی این عبارت، باید ابتدا عبارت موجود در داخلی ترین پرانتز را ارزیابی کرد. در این عبارت، ترتیب انجام عملیات به این صورت است: ابتدا عدد ۴ در ۲ ضرب می شود که حاصل آن ۸ است. بعد از ۲، عملگر تقسیم قرار دارد که عملوند بعدی آن در داخل پرانتز است. لذا تقدم عملگر + موجود در آن، از تقدم عملگر تقسیم بیشتر می شود و در نتیجه ۳ با یک جمع شده، حاصل آن ۴ است. اکنون، مقدار ۸ (که قبلاً محاسبه شد) بر ۴ تقسیم می شود و حاصل آن ۲ است. عملگر بعدی، + است ولی بعد از آن پرانتز آمده است. چون تقدم عملگرهای موجود در پرانتز بالاتر است، ابتدا ۲ از ۷ کم می شود که مقدار آن ۵ است و ۵ با ۶ جمع می شود که حاصل آن ۱۱ است و با مقدار ۲ که از قبل محاسبه شد جم می شود و حاصل عبارت ۱۳ است که در Y قرار میگیرد.

جدول: تقدم عملگرها در حالت کلی

()	بالاترین تقدم
! ~ ++ -- sizeof	
* / %	
+ -	
<< >>	
<<= >>=	
= = ! =	
&	
^	
&&	
?	
= += - = * = / = % =	پایین ترین تقدم

۶-۱. روش ایجاد برنامه

برای نوشتن برنامه در C، باید موارد زیر را در نظر بگیرید:

- ۱- تعیین نیازمندیهای مسئله
- ۲- تحلیل مسئله
- ۳- طراحی الگوریتم حل مسئله
- ۴- پیاده سازی الگوریتم
- ۵- تست و کنترل برنامه
- ۶- نگهداری و نوسازی برنامه

۶-۲. ۱- تحلیل مسئله

دانشجویی می خواهد با دانستن طول و عرض مستطیل، مساحت آن را حساب کند.

ورودی ها و خروجی های مسئله با استفاده از کلماتی که پررنگ شده اند مشخص می گردد:

ورودی های مسئله

- طول مستطیل
- عرض مستطیل

خروجی های مسئله

- مساحت مستطیل

وقتی ورودی ها و خروجی های مسئله مشخص شد، باید فرمول هایی تدوین شود که رابطه بین آنها را مشخص کند. فرمول محاسبه مساحت مستطیل به صورت زیر است:

$$\text{عرض مستطیل} \times \text{طول مستطیل} = \text{مساحت مستطیل}$$

در بعضی از موارد ممکن است رابطه بین ورودیها و خروجیها به این سادگی مشخص نشود و نیاز به فرضها و تسهیلات خاصی باشد. فرایند استخراج متغیرهای مسئله و تعیین روابط بین آنها از طریق صورت مسئله، انتزاع^۱ نام دارد.

^۱ - abstraction

۳-۶-۱. طراحی الگوریتم

در طراحی الگوریتم برای حل مسئله، لازم است قدم به قدم رویه‌هایی^۱ نوشته شوند - الگوریتم - و سپس بررسی شود که آیا الگوریتم، مسئله را به درستی حل می‌کند یا خیر. نوشتن الگوریتم، مشکل‌ترین بخش حل مسئله است. سعی نکنید تمام جزئیات مسئله را حل کنید، بلکه سعی کنید شیوه طراحی بالا به پایین^۲ را به کار ببرید. در روش طراحی بالا به پایین، ابتدا مراحل اصلی مسئله که باید حل شوند، مشخص می‌گردند و سپس با حل هر مرحله اصلی، کل مسئله حل می‌شود. اغلب الگوریتمها معمولاً مراحل زیر را دارا هستند:

۱- خواندن داده‌ها

۲- انجام محاسبات

۳- چاپ نتایج

وقتی مراحل مهم مسئله مشخص شدند، می‌توانید هر مرحله را به طور جداگانه حل کنید. به عنوان مثال، مرحله انجام محاسبات ممکن است به بخشهای کوچکتری تقسیم شود. این عمل را بهینه‌سازی الگوریتم گویند.

۹-۱. نگهداری برنامه

نگهداری و نوسازی^۳ برنامه شامل اصلاح برنامه جهت حذف خطاهای قبلی و نوسازی آن جهت پاسخگویی به نیازهای فعلی است. بعضی از سازمانها بعد از اینکه نویسنده برنامه‌ای به جایی دیگر منتقل شد، برنامه‌های آن را تا ۵ سال یا بیشتر نگهداری می‌کنند، ولی به تدریج آن را از دور خارج می‌کنند.

۱-۹-۱. فرآیند آماده‌سازی و اجرای برنامه

فرآیند آماده‌سازی و اجرای برنامه شامل موارد زیر است:

- ۱- وارد کردن برنامه در یک محیط ویراستاری و ذخیره کردن بر روی دیسک. فایل برنامه بر روی دیسک، در کامپایلرهای C++ با پسوند .cpp و در کامپایلرهای توربو C با پسوند .c ذخیره می‌شود.
- ۲- ترجمه برنامه جهت اشکالزدایی آن. اگر برنامه اشکالات نحوی^۴ داشته باشد، باید آنها را برطرف کرده و برنامه را دوباره ترجمه کنید. این مرحله را آنقدر انجام دهید تا کلیه اشکالات برنامه برطرف شوند.
- ۳- پس از ترجمه برنامه، فایلی با پسوند .obj ایجاد می‌گردد که به زبان ماشین است ولی قابل اجرا نیست. علتش این است که هنوز بخش‌های مختلف برنامه به هم پیوند نخورده‌اند و آدرس‌دهی آنها کامل نیست.
- ۴- پس از ترجمه برنامه، باید عمل پیوند^۵ را انجام دهید. در این مرحله، فایلی با پسوند .exe تشکیل می‌شود که قابل اجرا است.

1 - procedures

2 - top down design

3 - Update

4 - syntax error

5 - link

فصل دوم: ساختار برنامه C و ورودی - خروجی

بخشی از برنامه C باید کار اتصال فایل های سرآیند را به برنامه انجام دهد. برای این منظور از دستوری به نام `#include` استفاده می شود. این دستور که از دستورات پیش پردازنده است، معمولاً قبل از تابع `main()` قرار می گیرد. به طور کلی، باید به این نکته توجه داشته باشید که، پیش پردازنده، مترجمی است که با مشاهده دستوراتی که با `#` شروع می شوند، اجرا می شود و آنها را به دستورات زبان C تبدیل می کند. توجه داشته باشید که این دستور به `;` ختم نمی شود. 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 می شود:

`#include <stdio.h>`

در مورد دستور `#include` به نکات زیر توجه کنید:

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

۲-۱-۱. چاپ اطلاعات با تابع `printf()`

تابع `printf()` که در فایل `stdio.h` قرار دارد، برای چاپ اطلاعات در صفحه نمایش به کار می رود. اگر این تابع با موفقیت اجرا شود، تعداد کاراکترهایی را که به خروجی منتقل شده اند برمیگرداند و در صورت بروز خطا، یک عدد منفی را برمیگرداند. نحوه کاربرد این تابع به صورت زیر است:

`Printf("<عبارت ۲>, <عبارت ۱>")`

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

- ۱- اطلاعاتی که باید عیناً در خروجی چاپ شوند.
 - ۲- کاراکترهای تعیین کننده فرمت خروجی، این کاراکترها نوع اطلاعاتی را که در `<عبارت ۲>` ذکر شده اند و باید به خروجی برونده، مشخص میکنند. کاراکترهای فرمت با علامت `%` شروع می شوند. به عنوان مثال `%f`، برای چاپ اعداد اعشاری به کار می رود.
 - ۳- کاراکترهای کنترلی. این کاراکترها شکل خروجی اطلاعات را مشخص می کنند. 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 می شود. کاراکترهای کنترلی با `\` شروع می شوند. به عنوان مثال `\n` موجب میشود تا سطر جاری (سطری که فعلاً در حال نوشتن در آن سطر هستیم)، رد شود و چاپ اطلاعات از سطر جدیدی آغاز گردد.
- توجه داشته باشید که در تابع `printf()`، `<عبارت ۲>` می تواند وجود نداشته باشد. به عبارت دیگر، تابع `printf()` به صورت زیر نیز قابل استفاده است:

`Printf("<عبارت ۱>")`

۲-۱-۲. مشاهده صفحه خروجی برنامه

اگر برنامه هایی را که تاکنون در این فصل نوشته شد، در کامپیوترتان اجرا کرده باشید، دیدید که پس از تولید خروجی برنامه، کامپیوتر سریعاً به برنامه برمیگردد. لذا خروجی برنامه را نمی توانید مشاهده کنید. خروجی برنامه در صفحه ای غیر از صفحه ای که برنامه تان را تایپ می کنید تولید می شود. آن صفحه را صفحه خروجی می نامیم، اما خوب است که برنامه پس از تولید خروجی، در صفحه خروجی باقی بماند تا پس از مشاهده خروجی، برای برگشتن به برنامه، کلیدی فشار داده شود. برای این منظور

می توانید از تابع `getch()` استفاده کنید. این تابع منتظر فشردن کلیدی از صفحه کلید می ماند. 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 نمی کنید، پس از اجرای برنامه می توانید با کلید `ALT+F%` به صفحه خروجی بروید و پس از مشاهده اطلاعات، با فشردن کلید `Enter`، به صفحه برنامه برگردید.

۳-۲-۱. 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 نمی توانید 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 می شود. این تابع در فایل `conio.h` قرار دارد.

۴-۱-۲. انتقال مکان نما در صفحه خروجی

گاهی ممکن است بخواهید مکان نما را در صفحه خروجی 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 می شود. الگوی این تابع در فایل `conio.h` قرار دارد و به صورت زیر است:
`X` شماره ستون و `y` شماره سطر است که مکان نما به آنجا منتقل می شود. به عنوان مثال، دستور `gotoxy(40,10)` مکان نما را به ستون `۴۰` و سطر `۱۰` منتقل می کند.

۵-۱-۲. چاپ اعداد نوع `long , short`

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

مثال

برنامه ای که مقادیر `long` و `short int` را در خروجی نمایش می دهد.

```
#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;
}
```

۲-۲. ورودی و خروجی کاراکترها

همانطور که دیدید، با استفاده از توابع `scanf()` و `printf()` می توان ورودی و خروجی کاراکترها را انجام داد. ولی در `C`، توابع خاصی برای ورودی و خروجی کاراکترها منظور شد که کار کردن با آنها راحت تر از توابع `scanf()` و `printf()` است. در این بخش، تعدادی از آنها را مورد بررسی قرار می دهیم.

۲-۲-۱. خواندن کاراکتر با توابع getch() , getche()

این توابع، کاراکتری را از ورودی خوانده در متغیری قرار می دهند. این توابع در فایل conio.h قرار دارند و نحوه کاربرد آنها به صورت زیر است:

```
متغیر = getch();
```

```
متغیر = getche();
```

وقتی برنامه به این دستورات می رسد، منتظر می ماند تا کلیدی از صفحه کلید فشار داده شود. در این صورت، کاراکتر معادل آن کلید، در متغیر قرار می گیرد. این توابع به صورت زیر نیز قابل استفاده اند.

```
Getch();
```

```
Getche();
```

در این صورت، برنامه منتظر می ماند تا کلیدی از صفحه کلید فشار داده شود. پس از فشردن کلید، اجرای بقیه دستورات برنامه ادامه می یابد.

تابع getch() عکس عملی در صفحه نمایش ندارد. یعنی وقتی کلیدی فشار داده شد، کاراکتر معادل آن در صفحه نمایش ظاهر نمی شود. در حالی که تابع getche() پس از خواندن کاراکتر آن را در صفحه نمایش نیز ظاهر می کند. ضمناً در این توابع نیاز به فشردن کلید Enter نیست. در حالی که هنگام خواندن کاراکتر از طریق تابع scanf()، پس از وارد کردن کاراکتر، کلید Enter نیز باید فشار داده شود.

۲-۲-۲. خواندن کاراکتر با تابع getchar()

این تابع نیز همانند توابع getch() و getche() برای خواندن کاراکتر از صفحه کلید به کار می رود. این تابع در فایل stdio.h قرار دارد و نحوه کاربرد آن به صورت زیر است:

```
متغیر = getchar();
```

توجه داشته باشید که در این تابع، پس از ورود کاراکتر، باید کلید enter نیز فشار داده شود. این تابع کاراکتر خوانده شده را در صفحه نمایش نیز ظاهر می کند.

۲-۲-۳. وشتن کاراکتر با توابع putchar() و putch()

این توابع می توانند یک کاراکتر یا یک متغیر کاراکتری را در صفحه نمایش چاپ کند. تابع putchar() در فایل conio.h و تابع putchar() در فایل stdio.h قرار دارد و به صورت زیر به کار می روند:

```
Putch (متغیر);
```

```
Putch ('کاراکتر');
```

```
Putchar(متغیر);
```

```
Putchar('کاراکتر');
```

به عنوان مثال، دستورات زیر، کاراکتر 'a' را در خروجی چاپ می کند:

```
Putch ('a');
```

```
Putchar ('a');
```

مثال

برنامه ای که کاراکتری را از ورودی خوانده، با صدور پیامی آن را در خروجی چاپ می کند.

```
#include <conio.h>
```

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

```
Int main()
```

```
{
```

```
Char ch;
Clrchr();
Printf("\n Enter a character:");
Ch=getchar()
Printf(ch);
Getch();
Return 0;
}
```

خروجی

Enter a character: x
You typed the character : x

تحلیل مثال

اولین مرحله در حل مسئله این است که مشخص شود مسئله چه چیزی را خواسته است. باید یک سیستم اندازه گیری دمای هوا را به سیستم اندازه گیری دیگری تبدیل کنید. سیستم فارنهایت باید به سیستم سلسیوس تبدیل شود. لذا ورودی برنامه، درجه حرارت بر حسب فارنهایت است. شهروندان باید دما را بر حسب درجه سلسیوس بدانند. لذا خروجی برنامه، دمای هوا بر حسب درجه سلسیوس است. متغیر farendeg، محلی از حافظه را مشخص می کند که ورودی برنامه در آن قرار می گیرد (درجه فارنهایت) و متغیر centdeg، محلی از حافظه است که نتیجه محاسبه برنامه یا خروجی را نگه می دارد (درجه سلسیوس).

ورودی برنامه

ورودی برنامه، میزان درجه حرارت بر حسب فارنهایت است که در متغیر farendeg قرار می گیرد.

خروجی برنامه

خروجی برنامه، درجه حرارت بر حسب سانتیگراد است که در متغیر centdeg قرار می گیرد.

فرمول محاسبه

فرمول محاسبه درجه حرارت بر حسب سانتیگراد درجه حرارت بر حسب فارنهایت، به صورت زیر است:

$$\text{سلسیوس} = \left(\frac{5}{9}\right) * (\text{فارنهایت} - 32)$$

طراحی

اکنون الگوریتمی طراحی می کنیم که مسئله را حل کند:

الگوریتم

۱- درجه حرارت را بر حسب فارنهایت بخوان

۲- درجه حرارت را به سانتیگراد تبدیل کن

۳- درجه حرارت بر حسب سانتیگراد را نمایش بده.

اکنون باید مشخص کنید که هر مرحله از الگوریتم نیاز به اصلاحاتی دارد یا خیر. مرحله اول و مرحله سوم نیاز به اصلاح ندارند. مرحله دوم نیز واضح است، ولی اگر به جزئیات پرداخته شود، روشن تر می گردد. با توجه به رابطه بین درجه فارنهایت و درجه سلسیوس، الگوریتم را می توان به صورت زیر اصلاح کرد:

۱- درجه حرارت را بر حسب فارنهایت بخوان

۲- درجه حرارت را به سلسیوس تبدیل کن

۱-۲. رابطه بین فارنهایت و سلسیوس عبارت اند از: (۳۲-فارنهایت) * $\left(\frac{5}{9}\right)$ = سلسیوس

۳- درجه حرارت بر حسب سلسیوس را نمایش بدهو

اکنون الگوریتم را تست می کنیم. اگر درجه حرارت بر حسب فارنهایت برابر با ۵۰ باشد، با توجه به مرحله ۱-۲ الگوریتم، درجه حرارت بر حسب سلسیوس برابر است با:

$$\text{سلسیوس} = \left(\frac{5}{9}\right) * (50 - 32) = 10$$

در نتیجه، در مرحله سوم، عدد ۱۰ در خروجی چاپ می شود.

پیاده سازی

برای پیاده سازی الگوریتم، باید آن را به برنامه C تبدیل کنید. ابتدا باید به کامپایلر C بگویید که از چه ثوابت و متغیرهایی استفاده می کنید و سپس هر قدم الگوریتم را به یک یا چند دستور C تبدیل نمایید. برنامه و نمونه ای از اجرای آن در زیر آمده است.

```
#include <conio.h>
#include <stdio.h>
Int main()
{
Const int farconst = 32;
Float farendeg, centdeg;
Clrscr();
Printf("Enter farenheire.");
Scanf("%f", &farendeg);
Centdeg= ((float) 5/9) * (farendeg - farconst);
Printf("Celsius = %5.2f", centdeg);
Getch();
Return 0;
}
```

```
Enter farenheite : 70
Celsius : 21.11
```

خروجی

فصل سوم: حلقه های تکرار و ساختارهای تصمیم

۳-۱. ساختارهای تکرار

ساختارهای تکرار، تحت شرایط خاصی، یک یا چند دستور را چندین بار اجرا می کنند. به عنوان مثال، اگر بخواهیم تعداد ۱۰۰ عدد را از ورودی بخوانیم و آنها را با هم جمع کنیم. باید عمل خواندن عدد را ۱۰۰ بار تکرار کنیم. ساختارهای تکرار در زبانهای برنامه سازی مختلف به شکلهای گوناگونی مورد استفاده قرار میگیرند. این ساختارها را در زبان C مورد بررسی قرار می دهیم.

۳-۱-۱. ساختار تکرار for

ساختار تکرار for یکی از امکانات ایجاد حلقه است و معمولاً در حالتی که تعداد دفعات تکرار حلقه از قبل مشخص باشد، به کار می رود. در این ساختار، تغییری وجود دارد که تعداد دفعات تکرار حلقه را کنترل میکند. این متغیر را شمارنده یا اندیس حلقه تکرار می نامیم. اندیس حلقه دارای یک مقدار اولیه است و در هر بار اجرای دستورات حلقه، مقداری به آن اضافه می شود. این مقدار را که پس از هر بار اجرای حلقه به شمارنده اضافه می شود، گام حرکت گویند. گام حرکت می تواند عددی صحیح و اعشاری، مثبت یا منفی و یا کارا کتری باشد. یکی دیگر از اجزای حلقه for، شرط حلقه است. شرط حلقه مشخص می کند که دستورات داخل حلقه تا کی باید اجرا شوند. اگر این شرط دارای ارزش درستی باشد، دستورات داخل حلقه اجرا می شوند و گرنه کنترل برنامه از حلقه تکرار خارج می شود. اندیس حلقه تکرار می تواند عددی منفی، مثبت، صحیح و یا اعشاری و کارا کتری باشد. دستور for را به دو شکل می توان به کار برد:

روش اول	{ گام حرکت ; شرط حلقه ; مقدار اولیه اندیس حلقه } for
	دستور ۱
	دستور ۲
	دستور n
	}
روش دوم:	for (;;) {
	دستور ۱
	دستور ۲
	دستور n
	}

در هر یک از دو شیوه کاربرد، { می تواند در سطر بعدی (زیر for) قرار داشته باشد. ولی برای صرفه جویی در طول برنامه، در این کتاب به همین شکل که بیان شد استفاده می شود. در هر یک از روشهای کاربرد دستور for، چنانچه فقط یک دستور در حلقه وجود داشته باشد، نیازی به { و } نیست. در این حالت، این دستورات به صورت زیر قابل استفاده اند:

روش اول:	(گام حرکت ; شرط حلقه ; مقدار اولیه اندیس حلقه) for
	;دستور
روش دوم:	for (;;)
	; دستور

همان طور که ملاحظه می شود، در روش کاربرد دوم، for فاقد مقدار اولیه اندیس حلقه، شرط حلقه و گام حرکت است. این دستور برای ایجاد حلقه تکرار بی نهایت (حلقه تکراری که شرط پایان ندارد) مورد استفاده قرار می گیرد. برای خاتمه دادن به اجرای حلقه تکرار بی نهایت، باید کلید CTRL+BREAK را از صفحه کلید فشار داد.

۲-۳- حلقه های تکرار تودرتو

وقتی حلقه تکراری در داخل حلقه تکرار دیگر قرار داشته باشد. می گوییم که حلقه های تو در تو ایجاد شده اند. قانونی که بر حلقه های تکرار تو در تو حاکم است این است که، به ازای هر بار اجرای حلقه تکرار خارجی، حلقه تکرار داخلی به طور کامل اجرا می شود. ضمناً انتهای حلقه تکرار داخلی، زودتر از حلقه تکرار خارجی مشخص می شود. به عنوان مثال، در دستورات زیر، حلقه تکرار با اندیس i ، حلقه خارجی و حلقه تکرار با اندیس j ، حلقه تکرار داخلی است:

```
For (i=0; i<5; i++){
    ...
    For (j=0; j< 6; j++) {
        ...
    }
    ...
}
```

۳-۳. عملگر کاما و حلقه for

عملگر کاما (,) در فصل یک مورد بررسی قرار گرفت. این عملگر به حلقه for قابلیت انعطاف بیشتری می بخشد. با استفاده از این عملگر، می توان در قسمت مقدار اولیه حلقه و گام حرکت، دو یا چند عبارت را با هم ترکیب کرد. عبارات به ترتیب قرار گرفتن، و از چپ به راست ارزیابی می شوند. به عنوان مثال، دستور زیر را در نظر بگیرید:

```
For (I = 0, m += I, i < 10; i++, m++){
}
```

در این حلقه تکرار، اولین کاما، دو متغیر i و m را مقدار اولیه می دهد و دومین کاما، در هر تکرار، یک واحد به i و یک واحد به m اضافه می کند.

۴-۳. ساختار تکرار while

ساختار تکرار while یکی دیگر از امکاناتی است که برای تکرار اجرای دستورات به کار می رود. این ساختار به صورتهای زیر قابل استفاده است:

روش اول:	while (شرط) دستور ;
روش دوم:	while (شرط) { دستور ۱ دستور ۲ . . . دستور n }

همان طور که ملاحظه می کنید، وقتی دستورات تکرار شونده، بیش از یکی باشند، باید آنها را در بین { و } قرار داد. پس از اینکه اجرای برنامه به این دستور رسید، شرط حلقه تست می شود. اگر این شرط دارای ارزش درستی باشد، دستورات حلقه اجرا می شوند و گرنه کنترل برنامه از حلقه تکرار خارج می شود. برای اینکه حلقه خاتمه پیدا کند، شرط حلقه باید در داخل حلقه تکرار نقض شود. یعنی باید شرایطی در داخل حلقه فراهم شود تا شرط حلقه ارزش نادرستی پیدا کند و حلقه خاتمه یابد. اگر شرط حلقه

همیشه درست باشد(هیچگاه نقص نشود)، حلقه تکرار بی نهایت ایجاد می شود. در ادامه، مثالی را در این مورد مشاهده خواهید کرد.

مثال

برنامه ای که جمله ای را از ورودی خوانده، تعداد کاراکتر جمله را شمارش می کند. انتهای جمله به کلید Enter ختم می شود('r'). در این برنامه، count تعداد کاراکترهای ورودی است.

```
#include <stdio.h>
#include <conio.h>
Int main()
{
    Int count = 0;
    Clrscr();
    Printf("type a statement and ENTER to end:");
    While(getche() !='\r')
        Count + ++;
    Printf("\n length of statement is: %d", count);
    Getch();
    Return 0;
}
```

خروجی

Type a statement and ENTER to end: 1 learn C language.
Length of statement is : 19

۱-۴-۳. ساختار تکرار do ... while

ساختار تکرار do ... while مانند ساختار تکرار while است؛ با این تفاوت که در ساختار while، شرط حلقه در ابتدای حلقه تست می شود، در حالی که در do...while شرط حلقه در انتهای حلقه تست می گردد. بنابراین، دستورات موجود در حلقه do...while، در هر حال، حداقل یک بار اجرا می شوند.

۶-۳. ساختارهای تصمیم

همانطور که دیدید، ساختارهای تکرار، برای تکرار اجرای دستورات مورد استفاده قرار می گیرند. اما اگر بخواهیم، تحت شرایطی، تعدادی از دستورات اجرا شوند و یا تعدادی دیگر از دستورات اجرا نشود، باید از ساختارهای تصمیم استفاده کنیم. این ساختارها شرطی را تست کرده در صورت درست بودن شرط، مجموعه ای از دستورات را انجام می دهند. در زبان C چندین ساختار تصمیم وجود دارد که آنها را در این بخش بررسی می کنیم.

۱-۶-۳. ساختار تصمیم if

ساختار if که نام دیگرش، دستور انتقال کنترل شرطی است، شرطی را تست می کند و در صورتی که آن شرط دارای ارزش درستی باشد، مجموعه ای از دستورات را اجرا می کنند.

این دستور به صورت زیر به کار می رود:

```

    if (شرط)                                روش اول:
        دستور;
    Else
        دستور;
    if (شرط) {                               روش دوم:
        دستور ۱
        دستور ۲
        ...
        دستور n
    }
    Else {
        دستور n1
        دستور n2
        ...
        دستور n
    }

```

در هر یک از روشهای کاربرد، چنانچه شرط مورد بررسی درست باشد، دستور یا دستورات بعد از if و گرنه دستور یا دستورات بعد از else اجرا می شوند. اگر بیش از یک دستور بعد از if یا else بیایند، آن دستورات باید در بین { و } قرار گیرند. دستور if می تواند فاقد قسمت else باشد. در این صورت، چنانچه شرط مورد بررسی، درست باشد، دستورات بعد از if اجرا می شوند و گرنه بدون اجرای این دستورات، کنترل اجرای برنامه از if خارج می شوند.

مثال

برنامه ای که با خواندن یک جمله از ورودی، تعداد کاراکترها و کلمات موجود در جمله را شمارش می کند. کلمات با فاصله (space) از هم جدا شده اند و انتها به کلید Enter ختم می شود. متغیر charcount تعداد کاراکترها و متغیر wordcount تعداد کلمات جمله را شمارش می کند و ch کاراکتری است که از ورودی خوانده می شود.

```

#include <stdio.h>
#include <conio.h>
Int main ()
{
Int charcount = 0, wordcount = 0,
Char ch;
Clrscr();
Printf("\n Enter a statement(ENTER):");
While((ch = getch() != '\r'){
Charcount ++;
If(ch == ' ')
Wordcount ++;
} //end of while
Printf("\ncharcount= %d, wordcount = %d, charcount, wordcount+1)
Getch ();
Return 0;

```

}

خروجی

Enter a statement (ENTER): This book is my favourite.

Character count = 26, wordcount = 5

برای اطلاع از نحوه دریافت جزوات کامل با شماره های زیر تماس حاصل فرمایید.

۰۲۱-۶۶۹۰۲۰۶۱-۶۶۹۰۲۰۳۸-۰۹۳۷۲۲۲۳۷۵۶

خرید اینترنتی:

Shop.nokhbegaan.ir