فهرست مطالب

فصل ۱

اولین برنامه C++ شما

راهنمای کاربردی C++

چیزی نیست که به خاطر آن بخواهید از برنامه‌نویسی C++ بترسید- واقعاً راست می‌گویم! همانند هر زبان برنامه‌نویسی دیگری، این زبان نیز تنها برای دادن دستورالعمل‌های دقیق به یک کامپیوتر به کار می‌رود.

C++ می‌تواند به اندازه‌ای که می‌خواهید پیچیده شود اما روش آسان برای یاد گرفتن، استفاده از آن برای حل وظایف برنامه‌نویسی پایه‌ای است. این همان روشی است که در این جا به کار گرفته می‌شود.

در دو قسمت اول، من مفاهیم اولیه برنامه‌نویسی را مرور می‌کنم. اگر قبلاً از یک زبان برنامه‌نویسی دیگر استفاده کرده باشید، می‌توانید از این قسمت بگذرید. اما اگر تا به حال برنامه‌نویسی انجام نداده‌اید باید آن‌ها را مطالعه نمایید.

فکر کردن مانند یک برنامه‌نویس

ممکن است برنامه‌نویسی مانند فعالیت‌هایی که همیشه انجام می‌دهید نباشد. در اصل، شما دستورالعمل‌هایی را می‌دهید که این دستورالعمل‌ها باید از یک روش منطقی و سیستماتیک تبعیت کنند.

کامپیوترها کاری را انجام می‌دهند که به آن‌ها بگویید

کامپیوترها کاری را انجام می‌دهند که به آن‌ها می‌گوید: این مهم‌ترین قانون در این کتاب است، به خصوص اگر تازه شروع به برنامه‌نویسی کرده‌اید. با استفاده از یک زبان برنامه‌نویسی کامپیوتری، مانند C++، Visual Basic، Java یا FORTRAN، شما لیستی از کارهایی را به کامپیوتر می‌دهید که باید انجام دهد؛ این همان *برنامه* است.

البته یک کامپیوتر به اطلاعاتی نیز نیاز دارد که همان *داده‌های* ***(Data)*** برنامه هستند. همچنین کامپیوتر باید بداند که با این داده‌ها چه کاری باید انجام شود. دستورالعمل‌هایی که به کامپیوتر می‌گویند باید چه کاری انجام دهد، *کد* ***(Code)***برنامه نامیده می‌شوند.

تعیین کاری که برنامه باید انجام دهد

بنابراین، به منظور این که کامپیوتر کاری انجام دهد، باید به آن بگویید که دقیقاً باید چه کار کند.

احتمالاً قبلاً از یک کامپیوتر استفاده کرده و برنامه‌هایی که دیگران برای شما نوشته‌اند را اجرا کرده‌اید. در این حالت شما یک *کاربر نهایی* ***(End User)*** یا به صورت خلاصه *کاربر* هستید.

اما با نوشتن برنامه، خود را به سطح بالاتری در زمینه کامپیوتر ارتقا می‌دهید. اکنون شما تصمیم می‌گیرید که یک برنامه باید چه کاری انجام دهد و تعیین می‌نمایید که چه اتفاق‌هایی باید بیفتند.

اما یک کامپیوتر یک احمق کامل است. هیچ وقت نمی‌تواند حدس بزند که شما چه چیزی می‌خواهید. هیچ وقت نمی‌تواند قضاوت مستقلی انجام دهد. کامپیوتر کامل به چیزهایی که به آن گفته شده گوش داده و

همان کاری را انجام می‌دهد که شما می‌گوید. بنابراین باید مواظب منظوری که از حرف‌های شما به دست می‌آید باشید.

حتی نمی‌توانید دستوری به یک کامپیوتر بدهید که از نظر انسان‌ها کاملاً مشخص است، مانند "یک دما را از سلسیوس به فارنهایت تبدیل کن". بلکه باید کاملاً دقیق بوده و مراحل انجام کار خود را تعیین کنید:

۱. پیغام "دمای سلسیوس را وارد کنید:" را نمایش بده.

۲. یک عدد از صفحه کلید دریافت کرده و آن را در یک متغیر با نام ctemp ذخیره کن.

۳. با استفاده از فرمول ftemp=(ctemp * 1.8) + 32، دما را به فارنهایت تبدیل کن.

۴. پیغام "دمای فارنهایت برابر است با:" را نمایش بده.

۵. مقدار متغیر ftemp را نمایش بده.

اگر برای انجام یک کار ساده مانند این باید این همه زحمت کشید، پس چرا باید خود را اذیت کرد؟ پاسخ این است که وقتی یک برنامه نوشته شد، می‌توانید آن را بارها و بارها اجرا کنید و با این که نوشتن برنامه به زمان نیاز دارد، اما معمولاً با سرعت نور اجرا می‌شوند.

نوشتن دستورات معادل در ++C

بعد از این که دقیقاً تعیین کردید که برنامه باید چه کاری انجام دهد، باید دستورات ++C معادلی را برای هر مرحله بنویسید. یک دستور (Statement) یک معادل ++C برای یک جمله در زبان انگلیسی است.

برای مثال، می‌توانید از برنامه خود بخواهید تا کارهای زیر را انجام دهد:

۱. پیغام " :The Farenheit tempratrue is " چاپ شود.

۲. مقدار متغیر ftemp چاپ شود.

این مراحل را به این صورت به دستورات ++C تبدیل می‌کنید:

```
cout << "The Fahrenheit temperature is: ";
cout << ftemp;
```

به خاطر داشته باشید که هدف برنامه‌نویسی این است که کامپیوتر یک سری از وظایف مشخص شده را انجام دهد. اما کامپیوتر تنها زبان بومی خود را می‌فهمد: کد ماشین (Machine Code) که تنها حاوی صفر و ۱ است. در دهه ۱۹۵۰، برنامه‌نویس‌ها دستورات را با کد ماشین می‌نوشتند اما این کار بسیار سخت بوده و زمان زیادی لازم داشت.

برای آسان‌تر کردن کارها، مهندسان کامپیوتر زبان‌های برنامه‌نویسی مانند FORTRAN، Basic و C را ایجاد کردند که به انسان‌ها اجازه می‌دادند تا برنامه‌ها را با زبانی که قابل فهم‌تر برای انسان‌ها است بنویسند.

برای نوشتن یک برنامه، می‌توانید کار را با نوشتن یک *شبه دستورالعمل‌ها* **(Pseudocode)** شروع کنید- این روشی است که اغلب در این کتاب نیز استفاده می‌شود. شبه دستورالعمل‌ها مشابه با زبان عادی هستند اما عملی که برنامه انجام می‌دهد را با روشی سیستماتیک نشان می‌دهد که جریان منطقی برنامه را نمایش می‌دهد. برای مثال در این جا برنامه ساده‌ای را می‌بینید که یک شبه دستورالعمل را نشان می‌دهد:

اگر a بزرگ‌تر از b بود

عبارت " a بزرگ‌تر از b است" را چاپ کن

وگرنه

عبارت " a بزرگ تر از b نیست" را چاپ کن

وقتی شبه کد را نوشتید (که به آن الگوریتم نیز گفته می‌شود)، هنوز یک برنامه C++ ندارید. کاری که باید انجام دهید، پیدا کردن دستور منطبق با هر عمل در C++ است:

```
if (a > b)
     cout << "a is greater than b.";
else
     cout << "a is not greater than b.";
```

مزیت زبان برنامه‌نویسی در این است که از قوانینی تبعیت می‌کند که هیچ ابهامی در آن‌ها نباشد. دستورات C++ آنقدر دقیق هستند که می‌توان بدون نیاز به هیچ حدسی، آن‌ها را به کد ماشین تبدیل کرد.

نباید از داشتن چنین قوانین سختی برای برنامه‌نویسی تعجب کنید. این قوانین بسیار یکدست بوده و معمولاً ساده‌تر از قوانین در زبان انسانی هستند. بعضی وقت‌ها من این قوانین را به صورت خلاصه نشان می‌دهم. به عنوان مثال در این جا روش استفاده از یک if-else را می‌بینید:


```
if (condition)
     statement
else
     statement
```

کلمه‌هایی که به صورت توپر نشان داده شده‌اند، *کلمه‌های کلیدی* **(Keywords)** هستند؛ آن‌ها باید دقیقاً به شکل نشان داده شده در برنامه وارد شوند. کلمه‌های ایتالیک (کج شده) *جانگه‌دار* **(Placeholder)** هستند؛ آن‌ها نشان‌دهنده مواردی هستند که می‌توانید به دستور اضافه نمایید.

برنامه‌ای که دستورات C++ را به کد ماشین تبدیل می‌کند، یک *کامپایلر* **(Compiler)** *یا تفسیرکننده* نامیده می‌شود. در قسمت "ایجاد یک برنامه C++ " توضیح بیشتری درباره کامپایلرها می‌دهم اما در این جا چند تعریف اصلی را می‌بینید.

چند تعریف مهم

من سعی کردم تا از لغت‌های تخصصی سخت دوری کنم. اما وقتی می‌خواهید برنامه‌نویسی کامپیوتری را یاد بگیرید باید کلمه‌های جدیدی را نیز فرا بگیرید. در این جا چند تعریف که برای زندگی در این دنیا نیاز دارید را می‌بینید:

برنامه کاربردی (Application)

از نقطه نظر یک کاربر، برنامه کاربردی با یک برنامه (Program) یکسان است. اما یک برنامه کاربردی، برنامه‌ای است که کاربر اجرا کرده تا وظایف خاصی انجام شوند. به عنوان مثال یک واژه‌پرداز مانند مایکروسافت ورد، یک برنامه کاربردی است؛ و بنابراین اینترنت اکسپلورر نیز یک برنامه کاربردی است. حتی یک *کامپایلر* (که جلوتر تعریف می‌شود) نیز یک برنامه کاربردی است اما از نوع بسیار خاصی است زیرا توسط برنامه‌نویس‌ها استفاده می‌شود. برای ساده‌تر کردن موضوع، وقتی برنامه خود را نوشته، آن را ایجاد کرده و آزمایش کردید، یک برنامه کاربردی دارید.

کد (Code)

یک مترادف دیگر برای برنامه اما از نقطه نظر یک برنامه‌نویس. کاربرد عبارت *کد* از روزهایی شروع شد که برنامه‌نویس‌ها، برنامه‌های خود را با کد ماشین می‌نوشتند. هر دستورالعمل ماشین به یک ترکیب خاص از صفر و یک تبدیل شده و بنابراین برای انجام عملی خاص بود.

برنامه‌نویس‌ها صحبت درباره کد را ادامه داند حتی با این که از زبان‌های برنامه‌نویسی مانند C++، Java، FORTRAN یا Visual Basic استفاده می‌کردند. همچنین، عبارت کد برای متمایز کردن اطلاعات غیرفعال درون برنامه (داده‌های آن) از قسمتی که عمل‌هایی را انجام می‌داد (کد آن) استفاده می‌شود.

کامپایلر (Compiler)

مترجمی است که دستورات C++ (یعنی کد منبع C++) را به عنوان ورودی دریافت کرده و برنامه را به فرم زبان ماشین به شما می‌دهد. این کار لازم است زیرا پردازنده کامپیوتر تنها کد ماشین را درک می‌کند.

داده‌ها (Data)

اطلاعاتی که توسط یک برنامه استفاده می‌شوند. در ساده‌ترین حالت، این اطلاعات شامل کلمه‌ها یا اعداد هستند.

کد ماشین (Machine Code)

زبان بومی پردازنده‌های کامپیوتر که در آن هر دستورالعمل شامل ترکیبی منحصر به فرد (یا کد) از یک‌ها و صفرها است. می‌توانید یک برنامه را به صورت مستقیم به زبان ماشین بنویسید اما این کار نیازمند جستجوی هر دستورالعمل بوده و باید دانش کاملی درباره ساختار پردازنده‌ها داشته باشید.

زبان‌هایی مانند C++ روشی را فراهم آورده تا بتوانید برنامه‌ها را با ساختاری شبیه به زبان انگلیسی بنویسید که هنوز هم منطق آن به کد ماشین نزدیک است.

برنامه (Program)

یک سری دستورالعمل که توسط کامپیوتر به همراه داده‌های اولیه انجام می‌شوند. همان‌طور که قبلاً هم گفتم، نوشتن یک برنامه می‌تواند وقت‌گیر باشد، اما وقتی کامل شد معمولاً با سرعت زیادی انجام شده و می‌توان آن را بارها و بارها اجرا کرد.

کد منبع (Source Code)

یک برنامه که به زبان سطح بالایی مانند C++ نوشته شده و حاوی دستورالعمل‌ها به زبان مورد نظر می‌باشد. قبل از این که بتوانید برنامه را روی یک کامپیوتر اجرا کنید، ابتدا باید کد منبع به کد ماشین ترجمه شود. کد ماشین معمولاً در مبنای ۱۶ نمایش داده می‌شود و بنابراین چیزی مشابه این خواهد بود:

```
08 A7 C3 9E 58 6C 77 90
```

اما عملکرد این کد مشخص نیست. به جز این که دستورالعمل آن را جستجو کنید، درک چنین برنامه‌ای بسیار سخت است و به همین دلیل هیچ کس کدها را به زبان ماشین نمی‌نویسد. اما کد منبع تا حد زیادی مشابه با زبان انگلیسی است. به عنوان مثال دستور C++ بعدی می‌گوید:" اگر حقوق کم‌تر از صفر بود، پیغام خطایی را چاپ کن.":

```
if (salary < 0)
     print_error_message();
```

دستور (Statement)

یک دستور معمولاً معادل یک خط از برنامه C++ است و با یک نقطه ویرگول (;) خاتمه می‌یابد (مانند دستورات انگلیسی که با یک نقطه خاتمه می‌یابند.) اما C++ از دستورات ترکیبی نیز پشتیبانی می‌کند (مانند انگلیسی) و این دستورات می‌توانند چند خط باشند. اغلب دستورات C++ یک عمل را انجام داده در حالی که بعضی دیگر چند عمل را انجام می‌دهند.

کاربر

شخصی که یک برنامه را اجرا می‌کند- یعنی شخصی که از کامپیوتر برای انجام کاری مفید استفاده می‌کند مانند ویرایش یک فایل متنی، خواندن ایمیل، گردش در اینترنت یا بازی کردن. نام رسمی برای کاربر، *کاربر نهایی* ***(End User)*** است.

وقتی من در مایکروسافت بودم، کاربر شخصی بود که باعث اغلب مشکلات دنیا می‌شد اما در ضمن او شخصی بود که صورت‌حساب‌ها را نیز پرداخت می‌کرد. اگر چنین اشخاصی نباشند، دیگر شرکتی با نام

مایکروسافت نخواهد بود. بنابراین وقتی شروع به طراحی برنامه‌های جدی می‌کنید مهم است که نیازهای کاربر را نیز در نظر بگیرید.

چرا ++C متفاوت است؟

اغلب مواردی که درباره ++C گفته می‌شود درباره زبان‌های برنامه‌نویسی دیگر مانند پاسکال، FORTRAN، جاوا یا Basic نیز قابل استفاده هستند. همه آن‌ها زبان‌های برنامه‌نویسی *سطح بالا* ***(High Level)*** هستند، یعنی آن‌ها از زبان ماشین متفاوت بوده و از کلمه‌های کلیدی مانند if و while استفاده می‌کنند که به زبان انگلیسی نزدیک هستند.

هر زبان برای کاربرد خاصی ایجاد شده است. به عنوان مثال Basic به گونه‌ای طراحی شده است که یادگیری و استفاده از آن آسان باشد. هنوز هم مایکروسافت Visual Basic را به صورت ابزاری قدرتمند و سریع برای ویندوز ارائه می‌کند.

پاسکال برای استفاده در محیط‌های آکادمیک و به منظور آموزش مفاهیم برنامه‌نویسی استفاده می‌شود. پاسکال از Basic قدرتمندتر بوده اما به هیچ وجه با C یا ++C قابل مقایسه نیست. به علاوه، پاسکال عملاً دیگر از دور خارج شده است.

دانشمند اسطوره‌ای در زمینه کامپیوتر یعنی دنیس ریچی (Dennis Ritchie) کسی بود که C را برای کمک به نوشتن سیستم‌های عامل طراحی کرد. این یک زبان مناسب بود که از میانبرها پشتیبانی کرده و امکان نوشتن برنامه‌های فشرده‌تر را امکان‌پذیر می‌ساخت. روش استفاده سرراست ولی جامع C در بین بسیاری از برنامه‌نویس‌ها در سراسر دنیا محبوب شد.

اما ++C چه تفاوتی دارد؟

مهم‌ترین تفاوت بین C و ++C، توانایی استفاده از برنامه‌نویسی *شی‌گرا* ***(Object Oriented)*** در ++C است. این روش به خصوص برای سیستم‌های پیشرفته‌تر که حاوی عناصر پیچیده‌ای مانند رابط گرافیکی کاربر و محیط‌های شبکه هستند، بسیار مفید است. به عنوان یک برنامه‌نویس شی‌گرا شما سوالات زیر را خواهید پرسید:

۱. انواع داده اصلی (یعنی اطلاعات) در مسئله‌ای که باید حل شود، چی هستند؟

۲. چه عملیاتی باید برای هر نوع داده تعریف شود؟

۳. اشیا داده‌ها چگونه با دیگر اشیا برهم‌کنش و ارتباط دارند؟

در زمان یادگیری برنامه‌نویسی شی‌گرا، متوجه شدم که اگر ابتدا اصول برنامه‌نویسی را یاد گرفته باشید، کار راحت‌تر خواهد بود. بنابراین، تا فصل ۱۱ روی برنامه‌نویسی شی‌گرا متمرکز نخواهم شد.

اما من اشیا جدیدی - قطعه‌های اطلاعاتی که می‌توانند به عملکردهایی پاسخ دهند- را قبل از آن در کتاب معرفی خواهم کرد. برای مثال، در این فصل من از cout استفاده کردم که یک شی داده‌ای است که قسمتی از زبان C نیست. در C شما اطلاعات را با فراخواندن یک تابع چاپ می‌کردید که این تابع حاوی یک سری دستورات از پیش تعریف شده بود. اما وقتی از cout استفاده می‌کنید، داده‌ها ر به یک شی ارسال می‌کنید که *می‌داند چگونه* اطلاعات را نمایش دهد.

این یک روش عالی برای انجام کارها در اختیار قرار می‌دهد: cout می‌داند چگونه انواع بسیاری از اطلاعات را نمایش دهد. از همه بهتر، می‌توانید انواع داده که خودتان ایجاد کرده‌اید (یعنی کلاس‌ها) را بسط داده تا به خوبی با اشیا خروجی مانند cout کار کنند. انجام این کار در C به آسانی C++ نیست.

ایجاد یک برنامه C++

نوشتن یک برنامه، اولین مرحله ایجاد یک برنامه کاربردی است. ابتدا، باید دستورات برنامه را وارد نمایید.

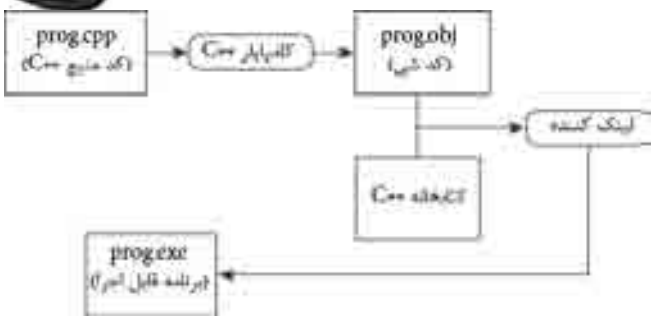
وارد کردن دستورات برنامه

برای نوشتن یک برنامه C++ باید روشی داشته باشید تا بتوانید دستورات برنامه را وارد کنید:

- می‌توانید از یک ویرایشگر متنی مانند Notepad در ویندوز استفاده کنید. اگر از چنین روشی استفاده می‌کنید، باید سند ایجاد شده را به صورت فایل متنی ساده (معمولاً گزینه Plain Text یا txt. در زمان ذخیره فایل) ذخیره نمایید.
- می‌توانید متن دستورات را در یک محیط طراحی یکپارچه (Integrated Development Enviroment یا IDE) وارد کنید. یک IDE در حقیقت یک ویرایشگر متنی است که با امکانات مخصوص برنامه‌نویسی مجهز شده است. به عنوان مثال Microsoft Visual Studio یک محیط طراحی است.

ایجاد برنامه (کامپایل و لینک کردن)

ایجاد یک برنامه فرآیند تبدیل کد منبع خود (دستورات C++) به یک برنامه کاربردی است. اگر از یک IDE استفاده می‌کنید، این کار به آسانی زدن یک کلید تابعی روی صفحه کلید است. این فرآیند در اصل حاوی دو مرحله است: کامپایل کردن و لینک کردن (که توابع استاندارد که برای شما نوشته شده‌اند را به برنامه وارد می‌کند). اگر این مراحل موفقیت آمیز باشند، آماده اجرای برنامه هستید.



اگر ایجاد برنامه با موفقیت انجام شود، می‌توانید خوشحال شوید؛ نه کامپایلر و نه لینک‌کننده خطایی پیدا نکرده‌اند. یعنی کار تمام شده است؟ خیر. این بدان معنی است که کامپایلر هیچ خطای گرامری (Syntax Error) یا خطای ساختاری در برنامه پیدا نکرده است. اما خطاهای زیادی هستند که کامپایلر نمی‌تواند پیدا کند. به عنوان مثال این وضعیت را در نظر بگیرید: فرض کنید جمله زیر را دارید:

ماه پنیر سبز ایجاد شده است.

از نظر گرامر این جمله دارای مشکل است. برای حل آن باید از کلمه *از* استفاده شود:

ماه *از* پنیر سبز ایجاد شده است.

اکنون این جمله از نظر گرامری خطایی ندارد. اما این بدان معنی است که جمله صحیح است؟ البته که خیر. لازم است که یک حرف *ن* به ابتدا *شده* اضافه کنید:

ماه از پنیر سبز ایجاد *نشده* است.

وضعیت مشابهی نیز در زبان‌های برنامه‌نویسی وجود دارد. کامپایلر ++C تعیین می‌کند که آیا برنامه حاوی خطای گرامری برای دستورات می‌باشد یا خیر و اگر خطایی داشته باشد، خطی که در آن خطا ظاهر شده است را گزارش می‌دهد. اما سوال بزرگ این است: *آیا برنامه همیشه در تمام موارد به درستی رفتار می‌کند.* پاسخ این سوال مشخص نیست و بنابراین مرحله بعدی ما بررسی این سوال است.

آزمایش برنامه

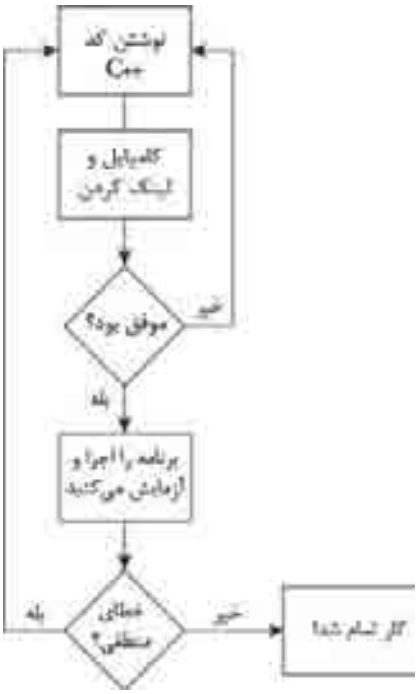
بعد از این که با موفقیت یک برنامه را ایجاد کردید، لازم است که آن را اجرا کنید تا ببینید آیا کار مورد نظر شما را انجام می‌دهد. در مورد یک برنامه واقعی- یعنی برنامه‌ای که فروخته شده و یا به دیگر افراد داده می‌شود- ممکن است لازم باشد آن را بارها و بارها آزمایش نمایید.

خطاهایی که در این مرحله به دنبال آن‌ها می‌گردید، *خطاهای منطق برنامه* ***(Program Logic Errors)*** نام دارند. این خطاها می‌توانند به مراتب بدتر از خطاهای گرامری باشند. فرض کنید برنامه اعداد اشتباهی چاپ

کرده یا بدون هیچ دلیل خاصی متوقف می‌شود. چه دستوری باعث این مشکل شده است؟ فرآیند آزمایش و تعیین منبع یک مشکل، *دیباگ کردن* ***(Debugging)*** نام دارد.

انجام بازنویسی‌های لازم

اگر برنامه به درستی اجرا شود، کار تمام است. اما اگر خطاهای منطقی وجود داشته باشد باید منبع خطا را مشخص کرده، تغییرات لازم را در برنامه اعمال کرده و آن را دوباره ایجاد کنید.



در نرم‌افزارهای پیچیده‌تر، لازم است چرخه بالایی را زیاد تکرار کنید. چنین برنامه‌ای ممکن است به تعداد زیادی آزمایش نیاز داشته باشد تا بتوان مطمئن شد که برنامه در تمامی حالت‌ها به خوبی عمل می‌کند. تا زمانی که کار آزمایش و بازنگری را تمام نکرده‌اید، برنامه به طور کامل ایجاد نشده است. اما در برنامه‌های ساده، می‌توان از تعداد کم‌تری آزمایش استفاده کرد.

نصب کامپایلر C++

وب سایت Pearson (در آدرس `www.informit.com/title/9780132673266`) حاوی یک لینک به کامپایلرهایی است که می‌توانید دانلود کنید مانند محیط Dev-C++ (که من آن را توصیه می‌کنم)؛ این

برنامه دارای یک محیط طراحی کامل است که می‌توانید برنامه‌های خود را در آن نوشته، کامپایل کرده و آزمایش نمایید.

مثال ۱,۱: چاپ یک پیغام

برای شروع به برنامه‌نویسی، یک فایل منبع جدید ایجاد کرده و کد زیر را در آن وارد کنید:

Print1.cpp

```
#include <iostream>
using namespace std;

int main() {
     cout << "Never fear, C++ is here!";
     system("PAUSE");
     return 0;
}
```

اگر از Visual Studio، Dev-C++ یا CodeBlock استفاده می‌کنید، انتظار نداشته باشید تا این کد به درستی کامپایل شود. برای جزئیات بیشتر به قسمت مربوط به IDEهای ذکر شده مراجعه کنید.

به خاطر داشته باشید که فاصله‌ها مهم نیستند اما C++ به بزرگی و کوچکی حروف حساس است.

اگر از یک IDE مانند Visual Studio یا Dev-C++ استفاده می‌کنید، کد بیشتری ظاهر خواهد شد که آن‌ها را رها کنید. همچنین ممکن است لازم باشد تا یک پروژه جدید را شروع کنید (در CodeBlock به این کار نیازی نیست).

اگر از محیط Dev-C++ استفاده می‌کنید، به منوی File رفته و New را انتخاب کنید و یک پروژه جدید را شروع کنید. سپس Console Application را به عنوان نوع پروژه انتخاب کرده و سپس یک نام مناسب برای پروژه وارد نمایید. سپس روی main.cpp در پنجره Project (سمت چپ صفحه) کلیک کنید. Dev-C++ کد زیر را برای شما ایجاد می‌کند:

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main() {
     system("PAUSE");
```

```
    return 0;
}
```

در این مورد، تنها کافی است دستوری که به cout شروع می‌شوند را وارد کنید؛ آن را در پایین خطی که int main(){ در آن قرار دارد و در بالای ;("system("PAUSE قرار دهید. بقیه خط‌ها را به همین صورت رها کنید.

اگر CodeBlock استفاده می‌کنید، لازم نیست تا یک پروژه ایجاد کنید تا بتوانید برنامه خود را اجرا نمایید. کافی است به منوی File رفته و سپس از منوی New مورد Empty File را انتخاب کنید. سپس در پنجره باز شده، C/C++ source را انتخاب کرده و روی Go کلیک کنید. در صفحات بعدی که ظاهر می‌شوند زبان برنامه‌نویسی C++ را انتخاب کرده و مسیر ذخیره فایل را تعیین کنید. در آخر روی Finish کلیک کنید تا فایل برنامه با نام مشخص شده باز شود.

روش دیگر انتخاب File->New->Empty File است که یک فایل خالی ایجاد می‌کند. سپس قبل از وارد کردن چیزی، فایل را با پسوند cpp. (برای C++) ذخیره می‌کنید تا آماده وارد کردن دستورات C++ باشد.

در این محیط، باید یک دستور دیگر در ابتدای فایل وارد کنید:

```
#include <cstdlib>
```

بنابراین کدی که باید وارد کنید به این صورت در می‌آید (دستور include اضافه شده به صورت توپر نمایش داده شده است):

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {
    cout << "Never fear, C++ is here!";
    system("PAUSE");
    return 0;
}
```

بعد از وارد کردن برنامه، آن را با نام print1.cpp ذخیره کرده، آن را کامپایل کرده و سپس اجرا نمایید. در این جا می‌بینید که برنامه بعد از کامپایل و اجرای صحیح چه چیزی در خط فرمان نمایش می‌دهد:

```
Never fear, C++ is here!
```

اگر از محیط Dev-C++ استفاده می‌کنید

اگر Dev-C++ را نصب کرده‌اید، در این جا چگونگی اجرای برنامه را می‌بینید:

۱. کلید F9 را بزنید تا برنامه ایجاد و اجرا شود.

۲. اگر هیچ پیغام خطایی دریافت نکردید، به این معنی است که برنامه با موفقیت کامپایل و لینک شده است. اگر پیغام خطایی دریافت کردید، یا کامپایلر را با موفقیت نصب نکرده‌اید یا قسمتی از مثال را به اشتباه تایپ کرده‌اید. دوباره به برنامه بازگشته و آن را با مثال نشان داده شده مقایسه کنید تا مطمئن شوید که کد را به درستی تایپ کرده‌اید.

اگر از محیط CodeBlock استفاده می‌کنید

اگر از CodeBlock استفاده می‌کنید، بعد از ذخیره کردن فایل، مراحل زیر را برای کامپایل و اجرای برنامه دنبال کنید:

۱. از منوی Build دستور Build را انتخاب کنید.

در پنجره Logs and Others، پیغام‌هایی مبنی بر کامپایل شدن فایل نمایش داده می‌شود. اگر فایل با موفقیت کامپایل و لینک شده باشد، برنامه نام فایل اجرایی ایجاد شده را به همراه اندازه آن نمایش می‌دهد و می‌گوید که فایل بدون هیچ خطا یا اخطاری ایجاد شده است. به عنوان مثال:

```
Compiling: D:\Documents\print1.cpp
Linking console executable: D:\Documents\print1.exe
Process terminated with status 0 (0 minutes, 0 seconds)
0 errors, 0 warnings
```

۲. از منوی Build دستور Run را انتخاب کنید.

یک خط فرمان باز شده که پیغام مورد نظر را نمایش می‌دهد. با زدن یک کلید روی صفحه کلید، پنجره بسته می‌شود.

اگر از Visual Studio استفاده می‌کنید

اگر Microsoft Visual Studio برای نوشتن برنامه‌های C++ خود استفاده می‌کنید باید چند کار دیگر نیز انجام دهید. Visual Studio ابزاری عالی برای نوشتن برنامه‌ها است اما برای ایجاد برنامه‌های ویندوز و نه برنامه‌های ساده طراحی شده است.

برای نوشتن یک برنامه در Visual Studio، ابتدا باید نوع مناسبی از پروژه را ایجاد کنید:

۱. از منوی File، زیرمنوی File را انتخاب کرده و سپس در آن دستور New را انتخاب کنید. یا اگر دکمه New Project در نزدیکی وسط صفحه نمایش داده می‌شود، روی آن کلیک کنید.

۲. نوع پروژه را Console Application انتخاب کنید. همچنین یک نام برای برنامه (در این مورد print1) وارد کرده و روی OK کلیک کنید.

۳. اگر فایل print1.cpp نمایش داده نشده است، در لیست نام‌های فایل در سمت چپ صفحه دنبال آن گشته و دو بار روی آن کلیک کنید.

قبل از این که کد C++ خود را وارد کنید، ابتدا تمام کدهایی که در فایل print1.cpp به جز مورد زیر می‌بینید را حذف کنید:

```
#include "stdafx.h"
```

باید این دستور در برنامه‌های کنسول (یعنی برنامه‌های تحت خط فرمان) که توسط Visual Studio ایجاد می‌شوند قرار داشته باشد. اگر از Visual Studio برای نوشتن برنامه‌های این کتاب استفاده می‌کنید، باید به خاطر داشته باشید که همیشه این خط را در ابتدای هر برنامه وارد کنید.

بنابراین برنامه print1.cpp باید مانند این باشد (خط مورد نظر به صورت توپر نشان داده شده است):

```
#include "stdafx.h"
#include <iostream>
using namespace std;

int main() {
     cout << "Never fear, C++ is here!";
     return 0;
}
```

برای ساختن برنامه، کلید F7 را بزنید. با این کار هم کامپایلر و هم لینک دهنده اجرا می‌شوند.

اگر برنامه با موفقیت ایجاد شد، کار تمام است. اما اگر ایجاد برنامه موفقیت‌آمیز نبود، به عقب بازگشته و مطمئن شوید که همه چیز به درستی انجام شده است.

برای اجرای برنامه، کلید Ctrl + F5 را بزنید. با این که روش‌های دیگری نیز برای اجرای برنامه از Visual Studio وجود دارد، اما این تنها روشی است که مانع از این می‌شود تا پنجره خط فرمان روی صفحه ظاهر شده و سریعاً ناپدید شود. فشردن Ctrl + F5 برنامه را اجرا کرده و سپس یک پیغام Press any key to continue را نمایش می‌دهد.

خط کد حاوی ;("PAUSE")syste لازم است تا پنجره کنسول سریعاً ناپدید نشود. با این حال اگر سیستم عامل شما ویندوز نیست، این دستور ممکن است کار نکرده و مجبور باشید آن را حذف کنید.

چگونگی عملکرد برنامه

باور کنید یا نه، این برنامه ساده تنها دارای یک دستور واقعی است. برای اکنون می‌توانید بقیه کد را به عنوان یک الگو در نظر بگیرید- یعنی چیزهایی که باید درون کد بوده اما می‌توانید با خیال راحت از آن‌ها چشم‌پوشی کنید.

در این جا موارد لازم استاندارد را به صورت توپر می‌بینید. برای اکنون نگران دلیل لازم بودن آن‌ها نباشید؛ تنها از آن‌ها استفاده کنید. در میان برکت‌ها ({})، خط‌های واقعی برنامه را قرار می‌دهید- که در این مورد تنها شامل یک دستور است.

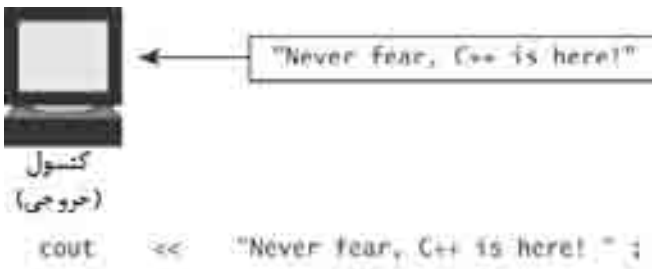
```
#include <iostream>
using namespace std;

int main() {
     Enter_your_statements_here!
     return 0;
}
```

این برنامه تنها دارای یک دستور واقعی است (که در خط پنجم برنامه قرار گرفته است):

```
cout << "Never fear, C++ is here!";
```

(به هیچ وجه نقطه ویرگول را فراموش نکنید) شاید سوال کنید که cout چیست؟ این مفهومی است که با جزئیات بیشتر در نیمه دوم این کتاب درباره آن توضیح می‌دهم. برای اکنون باید بدانید که cout مخفف Console Output است. به عبارت دیگر، نشان‌دهنده صفحه کامپیوتر می‌باشد. وقتی چیزی را به روی صفحه ارسال می‌کنید، آن چیز روی صفحه چاپ می‌شود.

در ++C، خروجی را با استفاده از cout و یک عملگر "جریان" (Stream) بعد از آن ایجاد می‌کنید (این عملگر نشان‌دهنده این است که جریان داده‌ها از یک مقدار- در این مورد عبارت Never fear, C++ is hear!- شروع می‌شود). می‌توانید آن را به صورت زیر تجسم کنید:



نقطه ویرگول (;) را فراموش نکنید. هر دستور ++C باید با یک نقطه ویرگول خاتمه یابد که البته چند استثنا وجود دارد.

به دلایل فنی، cout باید همیشه در سمت چپ مکانی قرار بگیرد که در آن‌جا استفاده می‌شود. در این مورد، داده‌ها به سمت چپ جریان پیدا می‌کنند. از فلش‌های سمت چپ که در حقیقت دو علامت کوچک‌تر (≪) هستند با یکدیگر استفاده می‌کنید.

جدول زیر چند کاربرد ساده cout را نشان می‌دهد:

دستور	عمل
cout ≪ "Do you C++?";	عبارت ?Do you C++ را چاپ می‌کند.
cout << "I think,";	عبارت ,I think را چاپ می‌کند.
cout << "Therefore I program.";	عبارت .Therefore I program را چاپ می‌کند.

تمرین‌ها

تمرین ۱،۱،۱. برنامه‌ای بنویسید که پیغام "Get with the program!" را چاپ کند. اگر میل داشتید، می‌توانید روی فایل منبع کار کرده و آن را تغییر دهید. (راهنمایی: تنها متن درون کوتیشن‌مارک‌ها "" را تغییر دهید.)

تمرین ۱،۱،۲. برنامه‌ای بنویسید که نام شما را چاپ کند.

تمرین ۱،۱،۳. برنامه‌ای بنویسید که عبارت ?Do you C++ را چاپ کند.

درباره #include و استفاده از آن

اگر به یاد داشته باشید من گفتم که خط پنجم برنامه، اولین دستور "واقعی" برنامه است. اما در اولین خط از متن زیر استفاده کردم:

```
#include <iostream>
```

این مثالی از یک راهنمای پیش‌پردازنده (Preprocessor Directive) است که یک دستور کلی برای کامپایلر C++ می‌باشد. یک راهنما (یا دایرکتیو) به فرم

```
#include <filename>
```

معرفی‌ها و تعریف‌هایی را بارگذاری می‌کند که قسمتی از کتابخانه استاندارد C++ را پشتیبانی می‌کنند. به عنوان مثال بدون راهنمای قبلی نمی‌توانستید از cout استفاده کنید.

اگر از نسخه‌های قدیمی‌تر C++ یا C استفاده کرده باشید، شاید تعجب کنید که چرا هیچ فایل خاصی (مانند یک فایل h.) مشخص نشده است. نام فایل یک *فایل*

اینکلود مجازی است که دارای اطلاعاتی به فرم پیش‌کامپایل شده می‌باشد.

اگر تازه به سراغ ++C آمده باشید، باید تنها به خاطر داشته باشید که از #include برای فعال کردن پشتیبانی از بخش‌های خاص کتابخانه استاندارد ++C استفاده کنید. بعداً وقتی شروع به استفاده از تابع‌های ریاضی مانند sqrt (برای ریشه مربعی یا جذر) کردید، می‌بینید که باید برای پشتیبانی از کتابخانه ریاضی از

```
#include <cmath>
```

استفاده کنید.

شاید بپرسید که چرا باید این کار اضافی انجام شود. فایل‌های اینکلود (Include) باعث می‌شوند بتوان بین زبان برنامه‌نویسی C و کتابخانه زمان اجرای استاندارد تمایز قائل شد. (بعضی وقت‌ها برنامه‌نویس‌های حرفه‌ای C/C++ از کتابخانه استاندارد استفاده نکرده و از کتابخانه خود استفاده می‌کنند.) توابع و اشیا درون کتابخانه همانند توابع تعریف شده توسط کاربر در نظر گرفته می‌شوند (که در فصل ۴ خواهید دید) که به این معنی می‌باشد آن‌ها باید معرفی شوند. به همین خاطر از فایل‌های اینکلود استفاده می‌شود.

همچنین لازم است که از یک دستور using نیز استفاده کنید. این دستور به شما اجازه می‌دهد تا به صورت مستقیم به اشیایی مانند std::cout مراجعه کنید. بدون این دستور، هر بار که می‌خواستید پیغامی را چاپ کنید، باید مسیر کامل cout را به این صورت مورد استفاده قرار می‌دادید:

```
std::cout << "Never fear, C++ is here!";
```

از cout (و پسر عموی آن cin) خیلی استفاده خواهیم کرد و بنابراین برای اکنون آسان‌تر است که در ابتدای هر برنامه از یک using استفاده کرده تا مجبور نباشید همیشه مسیر کامل شی را وارد نمایید.

رفتن به خط چاپ بعدی

وقتی متن‌هایی را توسط ++C در کنسول چاپ می‌کنید، متن به صورت خودکار به خط بعدی نمی‌رود. برای این کار باید یک کاراکتر *خط جدید* را چاپ کرده تا متن‌های بعد از آن به خط بعدی بروند. (استثنا: اگر هیچ وقت یک خط جدید را چاپ نکنید، وقتی متن خط کنونی را پر می‌کند به صورت خودکار به خط بعدی می‌رود اما اغلب این کار نتیجه‌ای با ظاهر نامناسب را ایجاد می‌کند.)

آسان‌ترین راه برای چاپ یک کاراکتر خط جدید استفاده از ثابت از پیش‌تعریف شده endl است. برای مثال:

```
cout << "Never fear, C++ is here!" << endl;
```

نام endl، مخفف end line است؛ بنابراین به صورت end ELL تلفظ می‌شوند نه به صورت end ONE. همچنین دقت کنید که endl در حقیقت به صورت std::endl است اما استفاده از using باعث می‌شود تا مجبور نباشید std:: را تایپ کنید.

روش دیگر برای چاپ کاراکتر خط جدید، استفاده از کاراکترهای \n است. این یک رشته فرار (Scape Character) است که C++ به جای در نظر گرفتن معنی تحت‌اللفظی آن، از معنای متفاوتی استفاده می‌کند. در این جا دستوری را می‌بینید که نتیجه مشابهی با مثال قبلی دارد:

```
cout << "Never fear, C++ is here!\n";
```

مثال ۱,۲: چاپ چند خط

برنامه این قسمت پیغام‌هایی را در چند خط چاپ می‌کند. همانند برنامه‌های قبلی، باید به خاطر داشته باشید که حروف کوچک و بزرگ به همان شکل مثال نشان داده شده وارد می‌شوند- با این که می‌توانید نوع حروف متن درون کوتیشن‌مارک‌ها را تغییر دهید.

print2.cpp

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {
     cout << "I am Blaxxon," << endl;
     cout << "the godlike computer." << endl;
     cout << "Fear me!" << endl;

     system("PAUSE");
     return 0;
}
```

این برنامه را با نام print2.cpp ذخیره کرده و آن را همانند مثال ۱,۱ کامپایل و اجرا کنید.

چگونگی عملکرد برنامه

این مثال مشابه با اولین مثالی است که دیدید. تفاوت اصلی در این مثال، به کار بردن کاراکترهای خط جدید است. اگر این کاراکترها حذف شوند، خروجی این برنامه به صورت زیر خواهد بود:

```
I am Blaxxon,the godlike computer.Fear me!
```

که برای ما مناسب نیست.

می‌توانید هر تعداد مورد را که می‌خواهید به این صورت چاپ کنید- تا وقتی که از یک کاراکتر خط جدید (endl) نرسیده باشید، نوشته‌ها درون یک خط چاپ می‌شوند.

می‌توانید با یک دستور، چندین مورد را به کنسول بفرستید:

```
cout << "This is a " << "nice " << "C++ program.";
```

وقتی برنامه حاوی این دستور اجرا می‌شود، متن زیر را در کنسول نمایش می‌دهد:

```
This is a nice C++ program.
```

یا حتی می‌توانید در دستور بالا، کاراکتر خط جدید را نیز قرار دهید، مانند این:

```
cout << "This is a" << endl << "C++ program.";
```

که خروجی زیر را دارد:

```
This is a
C++ program.
```

در این مثال، همانند مثال قبلی، از دستور return برای برگرداندن یک مقدار استفاده شده است. "برگرداندن یک مقدار"، فرآیند ارسال کردن یک سیگنال به عقب است- در این مورد به سیستم عامل یا محیط طراحی.

با استفاده از دستور return یک مقدار را بر می‌گردانید:

```
return 0;
```

مقدار برگشتی main، کدی است که به سیستم عامل ارسال می‌شود و در این جا صفر به معنی موفقیت است. مثال‌های این کتاب در اغلب موارد مقدار صفر را بر می‌گردانند.

تمرین‌ها

تمرین ۱,۲,۱. خطهای جدید را از مثال این قسمت برداشته اما فاصله بیشتری قرار دهید تا هیچ کدام از کلمات به یکدیگر نچسبند. (نکته: به خاطر داشته باشید که ++C به صورت خودکار بین متن‌های خروجی مختلف فاصله قرار نمی‌دهد.) نتیجه باید به این صورت باشد:

```
I am Blaxxon, the godlike computer. Fear me!
```

تمرین ۱,۲,۲. مثال را تغییر داده تا یک خط خالی را بین هر دو خط خروجی چاپ کند- به عبارت دیگر، دو برابر فاصله بین هر خط قرار دهد. (نکته: برای هر رشته متنی، دو کاراکتر خط جدید مورد استفاده قرار دهید.)

تمرین ۱,۲,۳. مثال را تغییر داده تا بین هر کدام از خط‌های خروجی، دو خط خالی چاپ شود.

رشته متنی (String) چیست؟

از ابتدای این کتاب، از متن‌هایی درون کوتیشن‌مارک استفاده کردم مانند این دستور:

```
cout << "Never fear, C++ is here!";
```

همه چیز خارج از این کوتیشن‌مارک‌ها از روش‌های نگارش C++ تبعیت می‌کنند. چیزی که درون کوتیشن‌مارک‌ها قرار دارد، داده‌های برنامه می‌باشد.

در واقعیت تمام داده‌های ذخیره شده توسط کامپیوتر، به صورت عدد ذخیره می‌شوند. اما بسته به چگونگی استفاده از داده‌ها، می‌توان آن‌ها را به صورت رشته‌هایی از کاراکترهای قابل چاپ در نظر گرفت. در این جا نیز همین وضعیت برقرار است.

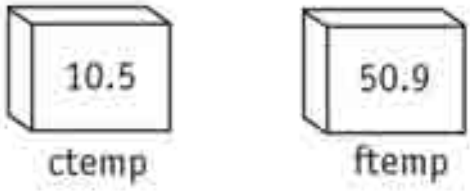
شاید اسم "کدهای اسکی (ASCII Code) " را شنیده باشد. این نوع داده "Never fear, C++ is here!" در این مثال است. کاراکترهای N، e، v، e، r و مانند آن‌ها به صورت بایت‌هایی ذخیره می‌شوند که هرکدام یک مقدار عددی منطبق با یک کاراکتر قابل چاپ هستند.

در فصل ۷، با جزئیات بیشتری درباره این نوع داده صحبت می‌کنم. نکته مهمی که باید در ذهن خود داشته باشید این است که داده‌های بین کوتیشن‌مارک‌ها به عنوان داده‌های خام در نظر گرفته می‌شوند که متفاوت از دستورها می‌باشد. این نوع داده یک *رشته از متن* ***(String of Text)*** در نظر گرفته شده که معمولاً با نام یک *رشته* ***(String)*** شناخته می‌شود.

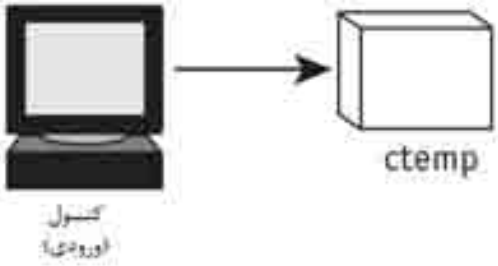
ذخیره کردن داده‌ها: متغیرهای C++

اگر تمام کاری که انجام می‌دادید، چاپ کردن پیغام‌ها بود، ++C اصلاً ابزار مناسبی نبود. نکته در این است که می‌توانید داده‌ها را از جایی – مانند ورودی کاربر- دریافت کرده و سپس کاری جالب با آن‌ها انجام دهید.

چنین کاری نیازمند استفاده از *متغیرها* ***(Variables)*** است؛ متغیرها مکان‌هایی هستند که داده‌ها درون آن‌ها نگه داشته می‌شوند. می‌توانید متغیرها را به عنوان جعبه‌های جادویی در نظر بگیرید که مقادیر درون آن‌ها نگه داشته می‌شوند. سپس می‌توان این مقادیر را خوانده، آن‌ها را تغییر داده یا دوباره آن‌ها را نوشت. مثال بعدی از متغیرهایی با نام ctemp و ftemp برای نگه داشتن مقادیر (به ترتیب) سلسیوس و فارنهایت استفاده می‌کند.



چگونه مقادیر درون متغیرها قرار می‌گیرند؟ یک روش از طریق ورودی کنسول است. در ++C، می‌توانید با استفاده از شی cin مقادیر را از ورودی دریافت کرده، و آن را به چیزی نشان دهید. با استفاده از cin از یک عملگر جریان استفاده می‌کنید که جریان داده‌ها را از چپ به راست نشان می‌دهد (<<):




```
cin      >>      ctemp ;
```

در این جا می‌بینید که در پاسخ به این دستور چه اتفاقی می‌افتد:

۱. برنامه متوقف شده و منتظر کاربر می‌ماند تا یک عدد را وارد کند.

۲. کاربر عدد را تایپ کرده و Enter را می‌زند.

۳. عدد پذیرفته شده و در این مورد، در متغیر ctemp قرار می‌گیرد.

۴. برنامه به اجرای خود ادامه می‌دهد.

بنابراین اگر درباره آن فکر کنید، در پاسخ به این دستور اتفاق‌های زیادی افتاده است:

```
cin >> ctemp;
```

اما قبل از این که بتوانید از یک متغیر در C++ استفاده کنید، ابتدا باید آن را معرفی کنید. این یک قانون مطلق است و باعث می‌شود C++ از زبان‌های برنامه‌نویسی آسان‌گیری مانند Basic متفاوت باشد که در این زمینه سفت نگرفته و نیازمند معرفی متغیر نیست.

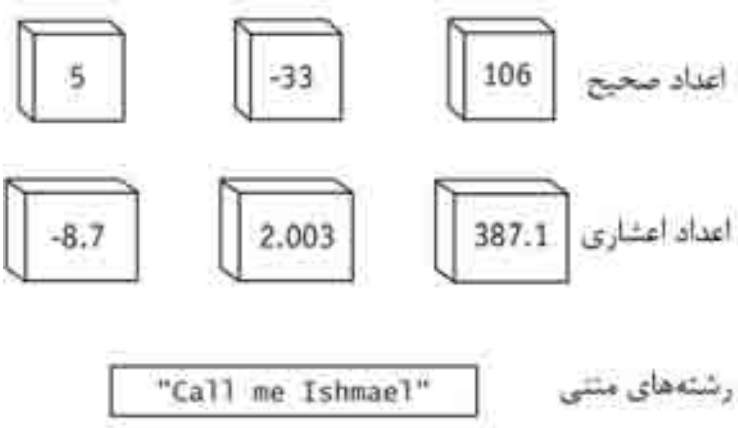
این یک قانون بسیار مهم است و باید همیشه آن را به خاطر داشته باشید:

در C++ باید قبل از استفاده از یک متغیر، آن را معرفی نمایید.

برای معرفی یک متغیر، ابتدا باید بدانید *نوع داده* ***(Data Type)*** آن چیست. این یک مفهوم مهم در C++ و اغلب زبان‌های برنامه‌نویسی است.

مقدمه‌ای بر انواع داده

بنابراین، یکم تغیر چیزی است که می‌توانید آن را به عنوان یک جعبه جادویی در نظر بگیرید که می‌توان اطلاعات – یا به عبارت دیگر داده‌ها- را درون آن قرار دارد. اما چه نوع داده‌هایی؟ تمام داده‌های روی یک کامپیوتر در نهایت به صورت عدد هستند اما به یکی از سه نوع اصلی تقسیم می‌شوند: اعداد صحیح (Integer)، اعداد اعشاری (Floating Point) و رشته‌های متنی (Text String).



تفاوت‌های مهمی بین نوع عدد صحیح و عدد اعشاری وجود دارد. اما قانون اصلی ساده است:

اگر باید اعداد با قسمت اعشاری را ذخیره کنید، از یک متغیر اعشاری استفاده می‌کنید؛ وگرنه از یک عدد صحیح استفاده می‌نمایید.

نوع داده اعشاری در C++ به صورت double است. ممکن است نام آن عجیب باشد؛ این نام مخفف Double Precision Floating Point (عدد اعشاری با دقت دو برابر است). یک نوع دیگر نیز با دقت عادی (نوع float) نیز وجود دارد اما به ندرت از آن استفاده می‌شود. وقتی باید قسمت‌های اعشاری را مورد استفاده قرار دهید، اگر از double استفاده کنید دقت نتایج بیشتر شده و خطای عددی نیز کاهش می‌یابد.

معرفی یک متغیر double به صورت زیر است. دقت کنید که این دستور با یک نقطه ویرگول خاتمه می‌یابد که کاملاً مشابه دستورات دیگر است:


```
double  variable_name;
```

در این جا double مشخص می‌کند که متغیر معرفی شده از نوع عدد اعشاری با دقت دو برابر است. variable_name نیز نامی است که به متغیر می‌دهید.

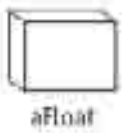
همچنین می‌توانید از معرفی double برای ایجاد یک سری از متغیرها از نوع double استفاده کنید. برای این کار double را وارد کرده و سپس نام متغیرها را لیست می‌کنید که هر نام متغیر توسط یک ویرگول از نام متغیر دیگر جدا می‌شود. به عنوان مثال:

```
double  variable_name1, variable_name2, ...;
```

برای مثال این دستور یک متغیر double با نام aFloat ایجاد می‌کند:

```
double  aFloat;
```

با این کار متغیر aFloat از نوع double ایجاد می‌شود:



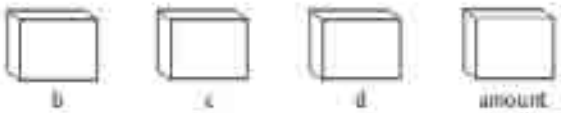
دستور بعدی، چهار متغیر b، c، d و amount را از نوع double معرفی می‌کند:

```
double  b, c, d, amount;
```

تاثیر این دستور معادل دستورات زیر می‌باشد:

```
double  b;
double  c;
double  d;
double  amount;
```

با این معرفی، چهار متغیر از نوع double ایجاد شده‌اند:



چرا دقت دو برابر بهتر از دقت معمولی است؟

دقت دو برابر محدوده بیشتری از مقادیر را پشتیبانی می‌کند و صحت بیشتری دارد: در حالی که دقت معمولی از ۴ بایت استفاده می‌کند، دقت دو برابر از هشت بایت استفاده می‌نماید.

هنگام انجام محاسبات، C++ تمام داده‌ها را به مقادیر double تبدیل می‌کند که منطقی است زیرا پردازنده‌های امروزی دارای یک پردازنده همکار ۸ بایتی هستند. به علاوه C++ ثابت‌های اعشاری را با دقت دو برابر ذخیره می‌کند مگر این که حالت دیگری را مشخص کنید (به عنوان مثال به جای 12.5 از 12.5F استفاده کنید).

دقت دو برابر دارای یک نقطه ضعف نیز می‌باشد: به حافظه بیشتری نیاز دارد. تنها وقتی تعداد زیادی مقادیر اعشاری را مورد استفاده قرار می‌دهید، این مورد مسئله مهمی می‌شود (به عنوان مثال در یک شبیه‌سازی با تعداد بسیار زیادی ذره). در این مواقع بهتر است به جای double از float استفاده کنید.

مثال ۱,۳: تبدیل واحدهای دما

هر بار که به کانادا می‌روم، باید دماهای سلسیوس را در ذهنم به فارنهایت تبدیل کنم. اگر یک کامپیوتر جیبی داشتم، خیلی خوب می‌شد که می‌توانست این تبدیل را برای من انجام دهد.

در این جا یک فرمول تبدیل را می‌بینید. وقتی از آستریکس (*) برای ترکیب کردن دو مقدار استفاده می‌شود، یعنی این دو مقدار در یکدیگر ضرب می‌شوند (به عنوان مثال a*b به معنای a ضربدر b می‌باشد).

```
Fahrenheit = (Celsius * 1.8) + 32;
```

اکنون یک برنامه مفید می‌تواند هر مقداری را برای دمای سلسیوس از کاربر دریافت کرده و آن را تبدیل نماید. این کار نیازمند استفاده از دو امکان جدید است:

- دریافت ورودی کاربر
- ذخیره مقدار ورودی در یک متغیر

در این جا برنامه کامل را می‌بینید. یک فایل منبع جدید باز کرده، کد را در آن وارد کرده و آن را با نام convert.cpp ذخیره کنید. سپس برنامه را کامپایل و اجرا نمایید.

convert.cpp

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {
     double  ctemp, ftemp;

     cout << "Input a Celsius temp and press ENTER: ";
     cin >> ctemp;
     ftemp = (ctemp * 1.8) + 32;
     cout << "Fahrenheit temp is: " << ftemp;

     system("PAUSE");
     return 0;
}
```

وقتی توضیحاتی نیز به برنامه اضافه کنید، خواندن برنامه آسان‌تر نیز می‌شود. این توضیحات (Comments) در ++C با یک جفت اسلش (//) شروع می‌شوند. به محض این که کامپایلر جفت اسلش را در هر جایی از خط برنامه می‌بیند، از آن جا تا انتهای خط را چشم‌پوشی می‌کند. بنابراین توضیحات هیچ تاثیری روی عملکرد برنامه ندارند اما برای کسانی که برنامه را می‌خوانند مفید هستند. در این جا نسخه‌ای را می‌بینید که دارای توضیحات است و این توضیحات به صورت پررنگ‌تر مشخص شده‌اند (توضیحات می‌توانند به هر زبانی باشند):

convert2.cpp

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {
     double ctemp;   // Celsius temperature

     double ftemp;   // Fahrenheit temperature

     // Get value of ctemp (Celsius temp).

     cout << "Input a Celsius temp and press ENTER: ";
     cin >> ctemp;
```

```
    // Calculate ftemp (Fahrenheit temp) and output.

    ftemp = (ctemp * 1.8) + 32;
    cout << "Fahrenheit temp is: " << ftemp << endl;

    return 0;
}
```

خواندن و درک این نسخه دارای توضیح برای انسان‌ها آسان‌تر است با این که به مقداری تایپ بیشتر نیاز دارد. در تمام مثال‌های این کتاب می‌توانید توضیحات را حذف کرده یا آن‌ها را تغییر دهید یا بعداً اضافه کنید. در هر صورت این قانون مهم را برای توضیحات به خاطر داشته باشید:

کامپایلر C++ از هر کدی که با // شروع شده باشد تا زمانی که به انتهای خط برسد چشم‌پوشی می‌کند.

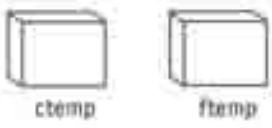
بنابراین اگر به بیش از یک خط توضیح نیاز دارید، هر خط توضیح را باید با یک // شروع نمایید. استفاده از توضیحات همیشه دلخواه است اما به کار بردن آن‌ها کار خوبی است به خصوص اگر قرار باشد برنامه را به شخص دیگری بدهید یا شاید بخواهید بعداً به برنامه خود مراجعه کرده و بتوانید از محتویات آن سر در بیاورید.

چگونگی عملکرد

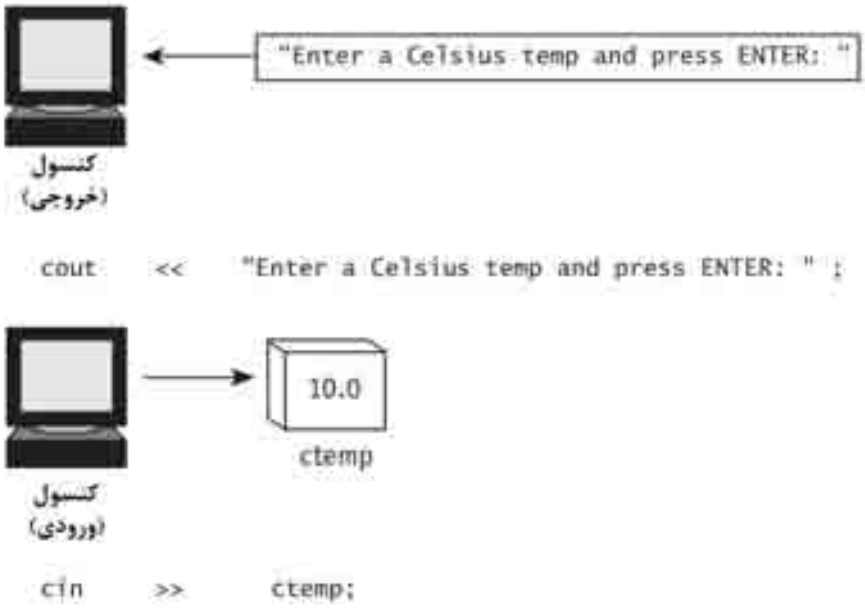
اولین دستور درون main، متغیرهایی را از نوع double با نام ctemp و ftemp معرفی می‌کند که به ترتیب برای ذخیره کردن دمای سلسیوس و فارنهایت به کار می‌روند:

```
double  ctemp, ftemp;
```

این دستور دو مکان برای ذخیره اعداد ایجاد می‌کند. به دلیل این که آن‌ها از نوع double معرفی شده‌اند، می‌توانند حاوی قسمت‌های اعشاری نیز باشند.



دو دستور بعدی به کاربر پیغام داده و سپس مقداری که کاربر وارد می‌کند را در متغیر ctemp ذخیره می‌کنند. فرض کنید کاربر مقدار ۱۰ را وارد می‌کند. سپس مقدار عددی 10.0 به متغیر ctemp نسبت داده می‌شود:



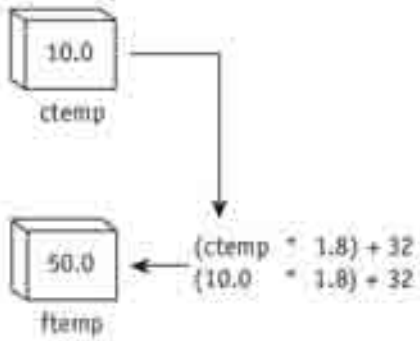
به صورت کلی شما از دستورات مشابهی در برنامه‌های خود استفاده کرده تا پیغامی را به کاربر نمایش داده و سپس ورودی او را ذخیره نمایید. این پیغام می‌تواند خیلی مفید باشد زیرا بدون آن کاربر نمی‌داند که باید چه کاری انجام دهد.

با این که عدد وارد شده در این مثال برابر با ۱۰ بود، اما به صورت 10.0 ذخیره می‌شود. در زبان ریاضی، 10 و 10.0 معادل هستند اما در زبان ++C، نگارش به صورت 10.0 نشان‌دهنده این است که مقدار به صورت اعشاری ذخیره می‌شود. اما 10 به این معنی است که مقدار به صورت عدد صحیح ذخیره می‌گردد. این روش عملکرد، نتایج مهمی نیز دارد.

دستور بعدی، کار تبدیل را انجام می‌دهد. برای این کار از مقدار ذخیره شده در ctemp برای محاسبه مقدار ftemp استفاده می‌شود:

```
ftemp = (ctemp * 1.8) + 32;
```

این دستور، یک عمل *نسبت‌دهی* ***(Assignment)*** را نشان می‌دهد. در این حالت مقدار سمت راست علامت تساوی (=) ارزیابی شده و سپس به درون متغیر در سمت چپ کپی می‌شود. این یکی از معمول‌ترین عملیات‌ها در ++C است. مجدداً با در نظر گرفتن این که کاربر مقدار ۱۰ را وارد کرده است، چگونگی جریان برنامه را می‌بینید:




```
ftemp = (ctemp * 1.8) + 32 ;
```

در آخر برنامه نتیجه را چاپ می‌کند که در این مورد برابر با ۵۰ است.


```
cout << "Fahrenheit temp is: " << ftemp ;
```

بهینه کردن برنامه

اگر با دقت به مثال قبلی نگاهی بیندازید، شاید از خودتان سوال کنید که آیا لازم بود دو متغیر معرفی شود در حالی که می‌توانستیم تنها یک متغیر را معرفی کنیم؟

در حقیقت این کار لازم نیست. به دنیای بهینه کردن برنامه‌ها خوش‌آمدید. نسخه بعدی این برنامه، نسخه اول را بهینه می‌کند. برای این کار متغیر ftemp حذف شده و عمل تبدیل و چاپ خروجی با یکدیگر ترکیب می‌شوند.

convert3.cpp

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {
```

```
    double  ctemp;   // Celsius temperature

    // Prompt and input value of ctemp.

    cout << "Input a Celsius temp and press ENTER: ";
    cin >> ctemp;


    // Convert ctemp and output results.

    cout << "Fahr. temp is: " << (ctemp * 1.8) + 32;
    cout << endl;

    system("PAUSE");
    return 0;
}
```

آیا الگوی کار را متوجه شدید؟ در این برنامه ساده، الگو به این صورت است:

۱. متغیرها معرفی شدند.

۲. بعد از نمایش یک پیغام، ورودی از کاربر دریافت می‌شود.

۳. محاسبات انجام شده و نتایج چاپ می‌شوند.

برای مثال، برنامه بعدی عمل متفاوتی را انجام می‌دهد که روش آن مشابه برنامه قبلی است. این برنامه به کاربر پیغام داده و تقاضای یک عدد می‌نماید. سپس توان دوی آن را چاپ می‌کند. دستورات آن مشابه با برنامه قبلی بوده اما از متغیر (n) و محاسبه متفاوتی استفاده می‌کند.

square.cpp

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main() {

    double  n;

    // Prompt and input value of n.

    cout << "Input a number and press ENTER: ";
    cin >> n;
```

```
    // Calculate and output the square.

    cout << "The square is: " << n * n << endl;

    system("PAUSE");
    return 0;
}
```

تمرین

تمرین ۱,۳,۱. مثال قبلی را به گونه‌ای بازنویسی کنید که تبدیل معکوسی را انجام دهد. یعنی یک مقدار فارنهایت برای متغیر ftemp دریافت شده و سپس به مقدار سلسیوس با متغیر ctemp تبدیل می‌شود. سپس نتایج چاپ می‌شوند. (نکته: فرمول تبدیل معکوس به صورت زیر است:

```
ctemp=(ftemp - 32) /1.8;
```

تمرین ۱,۳,۲. برنامه تبدیل فارنهایت به سلسیوس را تنها با یک متغیر ftemp بنویسید. این بهینه شده تمرین ۱,۳,۱ است.

تمرین ۱,۳,۳. برنامه‌ای بنویسید که یک مقدار را دریافت کرده و آن را به متغیر n داده و سپس مکعب آن (یعنی n*n*n) را در خروجی نمایش می‌دهد. مطمئن شوید که دستور خروجی به جای عبارت square از cube استفاده می‌کند.

تمرین ۱,۳,۴. مثال square.cpp را با استفاده از نام متغیر num به جای n بازنویسی کنید. مطمئن شوید که هر جایی n ظاهر می‌شود، نام متغیر را به درستی عوض می‌کنید.

چند کلمه‌ای درباره نام‌ها و کلمه‌های کلیدی

این فصل از متغیرهای ctemp، ftemp و n استفاده کرد. تمرین ۱,۳,۴ نیز پیشنهاد کرد که n را با num عوض کنید به شرطی که این کار در کل برنامه انجام شود. بنابراین num یک نام متغیر برای متغیر خواهد شد.

از نام‌های مختلفی می‌توانید برای متغیرها استفاده کنید. برای مثال می‌توانستید از نام‌هایی مانند killerRobot یا GovernorOfCalifornia استفاده کنید.

چه نام‌هایی برای متغیرها مجاز بوده و چه نام‌هایی غیرمجاز هستند؟ تا وقتی که از قوانین زیر تبعیت می‌کنید، می‌توانید از هر نامی که می‌خواهید استفاده نمایید:

- اولین کاراکتر باید یک حرف باشد و نمی‌توانید از یک عدد یا یک نشانه استفاده کنید. البته اولین کاراکتر می‌تواند خط زیر (_) باشد اما کتابخانه C++ از این قرارداد نام‌گذاری استفاده می‌کند و بنابراین بهتر است که از آن استفاده نکنید.
- بقیه نام می‌تواند به صورت حروف، اعداد یا خط زیر (_) باشد.
- نمی‌توانید در میان نام از فاصله استفاده کنید.
- باید از کلمه‌هایی که دارای معانی خاصی در C++ هستند استفاده کنید. به عنوان مثال نمی‌توانید از کلمه‌های کلیدی به عنوان نام متغیر استفاده نمایید.

البته لازم نیست که تمام کلمه‌های کلیدی C++ را به خاطر بسپارید. اگر نامی را مورد استفاده قرار داده باشید که کلمه‌های کلیدی جاوا تداخل دارد، کامپایلر یک پیغام خطا را نشان می‌دهد. در این حالت کافی است از یک نام متفاوت استفاده کنید. به علاوه اگر از یک برنامه IDE استفاده می‌کنید، معمولاً کلمات کلیدی با رنگ خاصی نمایش داده می‌شوند و بنابراین اگر نام متغیر شما با همان رنگ نمایش داده شد باید از یک نام دیگر استفاده کنید.

تمرین

تمرین ۱,۳,۵. در لیست بعدی، کدام کلمه‌ها نام‌های متغیر نامعتبر در C++ هستند و کدام یک معتبر می‌باشند:

x
x1
EvilDarkness
PennslyvaniaAve1600
1600PennsylvaniaAve
Bobby_the_Robot
Bobby+the+Robot
whatThe???
amount
count2
count2five
5count
main
main2

خلاصه فصل ۱

در این جا چند نکته اصلی فصل ۱ را می‌بینید:

- یک برنامه را با نوشتن کد منبع C++ شروع می‌کنید. این کد شامل دستورات C++ است که مشابه با زبان انگلیسی هستند. (اما کد ماشین کاملاً غیرقابل فهم است مگر این که معنی هر ترکیب صفر و یک را بدانید.) قبل از این که برنامه بتواند اجرا شود، باید به کد ماشین ترجمه شود تا تمام کامپیوترها آن را درک کنند.
- عمل ترجمه دستورات C++ به کد ماشین، کامپایل کردن نام دارد.
- بعد از کامپایل کردن، برنامه به توابع استاندارد ذخیره شده درون کتابخانه C++ لینک می‌شود. این فرآیند، لینک شدن نام دارد. بعد از انجام موفقیت‌آمیز این مرحله، یک برنامه اجرایی دارید که می‌توانید آن را به صورت مستقیم در سیستم عامل اجرا کنید.
- اگر از یک محیط طراحی یکپارچه (IDE) استفاده می‌کنید، فرآیند کامپایل و لینک کردن برنامه (ساختن آن) به صورت خودکار و با انتخاب یک دستور از درون برنامه انجام می‌شود.
- برنامه‌های ساده C++ دارای فرم کلی زیر هستند:

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main() {
     Enter_your_statements_here!
     return 0;
}
```

البته برای بعضی از کامپایلرها لازم نیست <cstdlib> include# را وارد کنید.

- برای چاپ خروجی، از شی cout استفاده می‌کنید. برای مثال:

```
cout << "Never fear, C++ is here!";
```

- برای چاپ خروجی و رفتن به خط بعدی، از شی cout استفاده کرده و درون آن یک کاراکتر خط جدید (endl) ارسال می‌کنید. برای مثال:

```
cout << "Never fear, C++ is here!" << endl;
```

- تقریباً همه دستورات C++ با یک نقطه ویرگول خاتمه می‌یابند.
- دو اسلش (//) مشخص کننده شروع یک توضیح است؛ تمام متن‌های بعد از آن تا انتهای خط توسط کامپایلر چشم‌پوشی می‌شوند. توضیحات برای کسانی که برنامه را می‌خوانند مفید است و به آن‌ها اجازه درک آسان‌تر عملکرد برنامه را می‌دهد.
- قبل از به کار بردن یک متغیر، ابتدا باید آن را معرفی کنید. برای مثال:

```
double  x;     //  متغیرxرا به عنوان یک عدد اعشاری با دقت دو برابر معرفی می‌کند
```

- متغیرهایی که اعداد از نوع اعشاری را در خود ذخیره می‌کنند باید از نوع double باشند. نوع ساده‌تر با دقت کم‌تر یعنی float تنها در زمانی که تعداد زیادی عدد اعشاری مورد استفاده قرار می‌گیرد، کاربرد دارد.
- برای دریافت ورودی صفحه کلید کاربر، از شی cin استفاده می‌کنید. برای مثال:

```
cin >> x;
```

- می‌توانید با استفاده از عملگر تساوی (=) مقداری را به یک متغیر نسبت دهید. با این کار مقدار مورد نظر درون متغیر قرار می‌گیرد. عملگر تساوی عبارت سمت راست خود را ارزیابی کرده و مقدار به دست آمده را به متغیر در سمت چپ نسبت می‌دهد. برای مثال:

```
x = y * 2;
```

در این جا مقدار y در ۲ ضرب شده و سپس نتیجه در x قرار می‌گیرد.