آموزش آسان و سریع

زبان C++

تالیف:

مهندس محمداسماعیلی‌هدی

انتشارات پندار پارس

سرشناسه : اسماعیلی هدی، محمد، ۱۳۵۶ –
عنوان و نام پدیدآور : آموزش سریع و آسان ++C/ تالیف محمد اسماعیلی‌هدی.
مشخصات نشر : تهران : پندار پارس، ۱۳۹۵.
مشخصات ظاهری : ۲۶۸ ص.: مصور،جدول.
شابک : 978-600-8201-19-9 : ۲۲۰۰۰۰ ریال
وضعیت فهرست نویسی : فیپا
موضوع : سی ++ (زبان برنامه‌نویسی کامپیوتر)
موضوع : C++ (Computer program language)
موضوع : زبان‌های برنامه‌نویسی کامپیوتر
موضوع : Programming languages (Electronic computers)
رده بندی کنگره : QA۷۶/۷۳ ۱۳۹۵ ۵الف۹۳س/
رده بندی دیویی : ۱۳۳/۰۰۵
شماره کتابشناسی ملی : ۴۳۹۱۷۲۸

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ www.pendarepars.com
تلفن: ۶۶۵۷۲۳۳۵ – تلفکس: ۶۶۹۲۶۵۷۸ همراه: ۰۹۲۱۴۳۷۱۹۶۴ info@pendarepars.com

نام کتاب : آموزش سریع و آسان ++C
ناشر : انتشارات پندار پارس
تالیف : محمد اسماعیلی هدی
چاپ نخست : مهر ماه ۹۵
شمارگان : ۵۰۰ نسخه
طرح جلد : رامین شکرالهی
چاپ، صحافی : روز
قیمت : ۲۲۰۰۰ تومان به همراه CD شابک : ۹۷۸–۶۰۰–۸۲۰۱–۱۹–۹

تقدیم به

معلمان گرامی

که همواره در اعتلای سرزمین ایران کوشیده‌اند.

فهرست مطالب:

دیباچه

به شکر الهی، تالیف و جمع‌آوری مطالب این کتاب پس از ماه‌ها کار و تلاش به پایان رسید و امید است مورد توجه و استفادهٔ دانشجویان گرامی و علاقه‌مندان به برنامه‌نویسی C++ قرار گیرد. کتاب حاضر، با توجه به نیاز دانشجویان کامپیوتر و دیگر رشته‌های دانشگاهی که در آن C++ تدریس می‌شود تدوین شده است. همچنین، این کتاب می‌تواند برای دانش‌آموزان هنرستانی و علاقه‌مندان به برنامه‌نویسی مورد استفاده قرار گیرد. در این کتاب با ایجاد درکی عمیق از برنامه‌نویسی سعی شده است تا یادگیری این زبان ساده‌تر و جذاب‌تر شود. هدف این کتاب، آموزش سریع و آسان C++ به همراه مثال‌های ساده و کاربردی است. برخی از نمونه‌های این کتاب از "C++ How to program" نوشته پائول و هاروی دیتل گرفته شده است.

این کتاب افزون بر آموزش C استاندارد (شامل ساختارها و دستورات پایه‌ای C)، به بررسی عمیق‌تر مفاهیم شئ‌گرایی (Object Oriented Programming) نیز پرداخته است. تمام برنامه‌های این کتاب به صورت کنسولی (Console Application) نوشته شده و از فرم‌های ویندوزی استفاده نشده است. برنامه‌های کنسولی، برنامه‌هایی هستند که واسط کاربری آنها بر پایهٔ متن است؛ بنابراین، در این برنامه‌ها، خبری از عناصر ویندوز مانند پنجره‌ها، جعبه‌های ویرایشی و دکمه‌های رادیویی نخواهد بود. یادگیری این کتاب می‌تواند مقدمه‌ای بر یادگیری زبان‌های Visual C++، C# و حتی ASP.NET (C#) باشد.

هنگام استفاده از این کتاب، به نکات زیر توجه کنید:

- هرگز کدها را از DVD کتاب کپی نکنید، بلکه آنها را تایپ کنید تا با خطاها و هشدارهای کامپایلر بیشتر آشنا شوید. کدهای همراه کتاب، زمانی کارآمد هستند که با نوشتن آنها به نتیجه نرسیده باشید. همچنین، پس از نوشتن مثال‌های کتاب، اعداد و دستورات آن را تغییر دهید تا تجربیات جدیدتری کسب نمایید.
- در صورت برخورد با خطاها، آنها را خودتان برطرف کنید و تا حد امکان از دیگران کمک نگیرید.
- پس از یادگیری هر بخش، برنامه‌های کوتاه مورد نظر خود را بنویسید. برای یادگیری یک زبان برنامه‌نویسی، چیزی بهتر از تمرین و نوشتن برنامه‌های گوناگون وجود ندارد.

در نهایت، مثال‌های این کتاب اشکال‌زدایی شده و بدون خطاهای گرامری هستند. امید است ایرادهای باقیمانده در کتاب نیز با نظرات و همیاری اساتید محترم و دانشجویان گرامی رفع شده و با کیفیت بهتر در دسترس عزیزان قرار گیرد. اساتید محترم و دانشجویان گرامی می‌توانند نظرات خود را از طریق انتشارات پندارپارس یا پست الکترونیکی mohammadesmailihoda@gmail.com در اختیار اینجانب قرار داده و ما را در ارائهٔ هر چه بهتر مطالب یاری رسانند.

محمداسماعیلی‌هدی – مهرماه ۱۳۹۵

فصل نخست

آشنایی با C++

زبان C استاندارد، در سال ۱۹۷۲ توسط دنیس ریچی برای نوشتن برنامه‌های سیستمی توسعه پیدا کرد و برای تولید سیستم‌عامل‌های گوناگون مورد استفاده قرار گرفت. پیش از توسعهٔ این زبان، از اسمبلی برای نوشتن برنامه‌ها و سیستم‌عامل‌ها استفاده می‌شد که کار برنامه‌نویسی را بسیار سخت می‌کرد. در سال ۱۹۷۹ مجموعه‌ای از قابلیت‌های جدید به C افزوده شد و در سال ۱۹۸۳ نام C++ را به خود گرفت. هم اکنون، از C++ و زبان‌هایی مانند C# که بر پایهٔ آن ساخته شده‌اند، برای نوشتن بسیاری از برنامه‌های کامپیوتری استفاده می‌شود. همچنین، زبان C استاندارد، هنوز برای نوشتن برنامه‌های سخت‌افزاری کاربرد دارد.

زبان C++ یکی از زبان‌های سطح میانی است که می‌تواند برای نوشتن برنامه‌های کاربردی و سیستمی مورد استفاده قرار گیرد. یکی از مزایای بزرگ این زبان، هستهٔ مرکزی کوچک و قابل حمل آن است[۱]. این هسته، شامل گرامر زبان و دستورات پایه‌ای آن است. سایر دستورات و توابع زبان در کتابخانه‌های آن قرار دارند. برای استفاده از این دستورات، تنها باید کتابخانه‌های مورد نیاز را به برنامهٔ خود ملحق کنید. این امر انعطاف‌پذیری زیادی به زبان C++ داده و باعث می‌شود تا بتوانید از میان کتابخانه‌های متنوع، کتابخانهٔ دلخواه خود را انتخاب کنید. ساخت‌یافتگی[۲] زبان C++، یکی دیگر از قابلیت‌های این زبان محسوب می‌شود. ساخت‌یافتگی و شئ‌گرایی[۳]، امکان تولید برنامه‌های بزرگ و مدیریت آن را فراهم می‌کنند. ایجاد برنامه‌های کم حجم و سریع، یکی دیگر از ویژگی‌های کامپایلر[۴] C++ است که باعث کارآمدی برنامه‌های تولید شده با این زبان می‌شود. شکل روبه‌رو، این قابلیت‌ها را به صورت یک نمودار نشان می‌دهد.

---

[۱] منظور از قابل حمل بودن، امکان انتقال برنامه‌ها بین سیستم‌عامل‌های گوناگون است که با کمترین تغییرات امکان‌پذیر است.

[۲] ساخت‌یافتگی، امکان تقسیم برنامه به بخش‌های کوچکتری به نام تابع یا روال را فراهم می‌آورد.

[۳] در این مدل، برنامه‌ها از اشیا گوناگون ساخته می‌شوند. یک شئ، از چند متد (تابع) و متغیرهای وابسته به آن تشکیل می‌شود.

[۴] کامپایلر، برنامه‌ای است که کدهای نوشته شده به زبان C++ را به یک برنامهٔ اجرایی قابل درک به وسیلهٔ پردازنده تبدیل می‌کند. یکی از وظایف کامپایلر، پیدا کردن خطاهای برنامه و اعلان آن است.

یک برنامه (نرم‌افزار)، مجموعه‌ای از دستورات است که به کامپیوتر داده می‌شود تا به او بگوید چه کاری باید انجام دهد. کامپایلر C++ دستورات را دریافت کرده و آن را به زبان ماشین (قابل درک به وسیلهٔ پردازنده) تبدیل می‌کند. در حالت کلی، زبان‌های برنامه‌نویسی مانند C++، ما را از نوشتن برنامه‌ها به زبان ماشین، می‌رهاند.

در این فصل، مطالب زیر را خواهید دید:

- ✓ ساختار کلی برنامه‌های C++
- ✓ استفاده از توضیحات در برنامه
- ✓ دستورات ورودی و خروجی
- ✓ مراحل نوشتن برنامه و خطایابی آن

۱.۱. نوشتن و اجرای برنامه‌ها

برنامه‌های زبان C++ در هر ویرایشگر متنی (از جمله Notepad) قابل نوشتن است؛ هرچند، بهتر است از ویرایشگرهای مربوط به کامپایلرهای C++ استفاده نمایید تا امکانات ویژه‌ای در مورد نوشتن، ویرایش و خطایابی کدها در اختیار شما قرار دهند. برنامه‌های زبان C++ با پسوند cpp.* و برنامه‌های زبان C با پسوند c.* ذخیره می‌شوند.

برای آن که کامپیوتر دستورات برنامه را انجام دهد، باید آنها را در یک کامپایلر مناسب، خطایابی، ترجمه و سپس اجرا کنید. نحوهٔ انجام این کار در کامپایلرهای گوناگون، متفاوت است. برای اجرای برنامه‌ها در Boland C++ از ترکیب کلیدهای Ctrl+F9 و در Visual Studio از ترکیب کلیدهای Ctrl+F5 استفاده می‌شود.[۱] معمولا با اجرای برنامه‌ها، یک فایل اجرایی تولید می‌شود. این گونه فایل‌ها، مستقل از کامپایلر و در سیستم‌عامل ویندوز قابل اجرا هستند.

پس از نوشتن برنامه‌ها و خطایابی در محیط ویرایشگر C++، کامپایلر یک فایل Object با پسوند obj.* از برنامهٔ شما می‌سازد. این فایل در نهایت به وسیلهٔ Linker به یک برنامهٔ اجرایی تبدیل می‌شود.

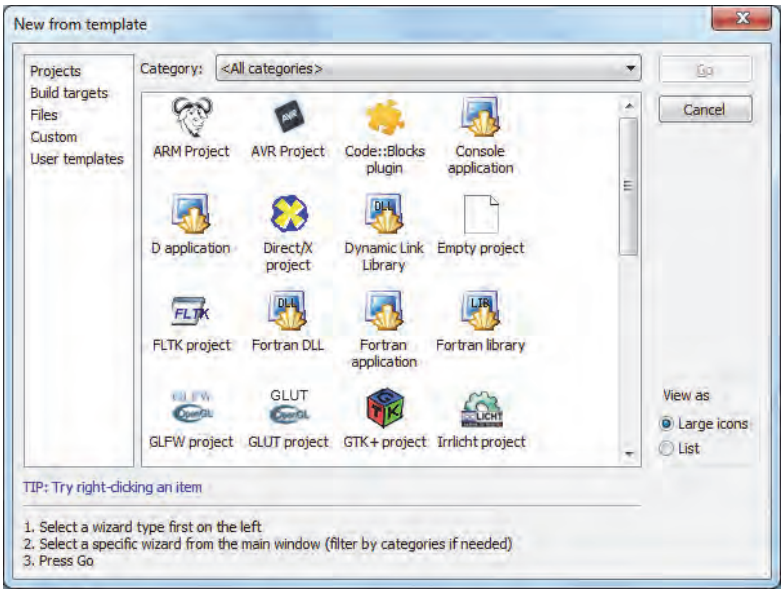
در میان انواع کامپایلرها، استفاده از CodeBlocks به دلیل داشتن محیط‌های گوناگون می‌تواند بسیار مفید باشد. در این بخش، به بررسی فشردهٔ این کامپایلر خواهیم پرداخت. فایل نصبی این برنامه، به همراه چند کامپایلر دیگر در DVD همراه کتاب قرار داده شده است.

برای نوشتن برنامه‌ها و اجرای آنها در محیط CodeBlocks باید مراحل زیر را انجام دهید:

۱. منوی File ➤ New را باز کرده و یا روی آیکن کلیک کنید.
۲. از لیست باز شده، یکی از گزینه‌های زیر را انتخاب کنید:

[۱] برای ترجمهٔ کدهای C++ می‌توانید از برنامه‌های Dev CPP، Free C++، Turbo C++ و CodeBlocks نیز استفاده کنید.

Empty file: یک فایل خالی در اختیار شما قرار می‌دهد. این فایل می‌تواند حاوی هر نوع کـدی باشـد؛ بنـابراین، موقع ذخیره کردن آن باید پسوند فایل را مشخص کنید.
File: با انتخاب این گزینه می‌توانید یک فایل جدید برای نوشتن برنامه‌های خود ایجاد کنید. پس از انتخـاب ایـن گزینه، باید نوع فایل و محل ذخیره کردن آن را مشخص نمایید.
Project: در این بخش می‌توانید یکی از پروژه‌های تعریف شده در برنامهٔ CodeBlocks را انتخاب کنید.



برای نوشتن برنامه‌های C یا ++C موجود در این کتاب، باید گزینـهٔ Console application را انتخـاب کـرده و دکمهٔ Go را فشار دهیـد. در ادامـه، بایـد زبـان مـورد نظـر و نـام پـروژه را مشـخص نماییـد. سـپس، در بخـش Management در برگهٔ Projects، پروژهٔ مورد نظر را باز کرده و روی main.cpp دابل کلیک کنید.

۳. در پنجرهٔ باز شده، کدهای مورد نظر را تایپ کنید.
۴. برای خطایابی و اجرای برنامه، روی یکی از آیکن‌های زیر کلیک کنید[۱]:

Build برای کامپایل برنامه استفاده می‌شود. کلید میانبر Ctrl+F9 نیز همین کار را برای شما انجام می‌دهد.
Run برای اجرای برنامهٔ کامپایل شده، از این آیکن استفاده کنید. برای این کار، از کلید میانبر Ctrl+F10 نیـز استفاده می‌شود.
Build and run برای کامپایل و اجرای همزمان برنامه به کار می‌رود. کلید میانبر این دستور F9 است.

۵. در صورت لزوم، برای اجرای خط به خط برنامه می‌توانید از دستورات منوی Debug استفاده کنید.
۶. اگر به طور همزمان از پروژه استفاده می‌کنید، برای مدیریت آنها باید به سراغ پنجرهٔ Projects رفته و روی پروژهٔ مورد نظر کلیک راست کنید. سپس، یکی از گزینه‌های زیر را انتخاب کنید:

---

[۱] این دستورات، در منوی Build نیز در دسترس شما قرار دارند.

Save project: از این گزینه، برای ذخیره کردن تغییرات پروژه استفاده کنید.
Close project: این دستور، برای بستن پروژه استفاده می‌شود.
Activate project: برای فعال کردن یک پروژه استفاده می‌شود. برنامهٔ CodeBlocks، تنها پروژهٔ فعال را خطایابی، کامپایل یا اجرا می‌کند.
Add files: برای اضافه کردن فایل‌های جدید به پروژه مورد استفاده قرار می‌گیرد.
Format this project (AStyle): برای تنظیم تورفتگی‌های برنامه و قالب‌بندی آن به کار می‌رود.
Clean: برای پاک کردن فایل‌هایی که در هنگام کامپایل برنامه ایجاد می‌شوند، به کار می‌رود. در صورت استفاده از این دستور، برنامه باید پیش از اجرا دوباره کامپایل شود.
Build options: تنظیمات مربوط به نحوهٔ کامپایل برنامه، در این بخش قرار داده شده است.
Open Project Folder in File Browser: برای مشاهدهٔ محتویات فولدر پروژه در برگهٔ Files به کار می‌رود.

۱.۲. ساختار یک برنامه

در اینجا قصد داریم ساختار کلی یک برنامه C++ را که در همهٔ برنامه‌ها دیده می‌شود بررسی نماییم. دقت کنید که کامپیوتر یک موجود هوشمند نیست؛ بنابراین، باید ساختار و قوانین برنامه‌نویسی را دنبال کنید. وگرنه، برنامه‌ها دارای خطا بوده و اجرا نخواهند شد.

به نمونه برنامهٔ زیر توجه کنید: (Chapter 1\first.cpp)

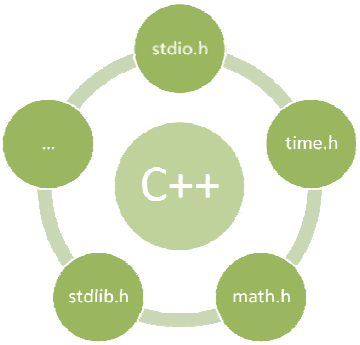
```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello C++ World!";
   // Any variables and codes.
   return 0;
}
```

```
Output:

Hello C++ World!
```



خط نخست، فایل iostream را به برنامهٔ بالا ملحق می‌کند؛ به این ترتیب می‌توان از دستورات موجود در این کتابخانه استفاده نمود. اتصال کتابخانه‌های گوناگون به برنامه و استفاده از آنها، باعث انعطاف‌پذیری بالای برنامه‌های C++ و افزایش قابلیت‌های آن می‌شود. نمودار روبه‌رو برخی دیگر از کتابخانه‌های استاندارد زبان C++ را نشان می‌دهد. می‌توانید این کتابخانه‌ها را با کمک دستور #include به برنامه‌های خود اضافه کرده و از قابلیت‌های آنها استفاده کنید. فایل‌های کتابخانه (Header Files) معمولا دارای پسوند h.* هستند. دستور using namespace در خط دوم مشخص می‌کند که دستورات برنامه در

فضای کاری استاندارد تعریف شده‌اند. این خط دستوری، در کامپایلرهای قدیمی‌تر وجود ندارد؛ بنابراین، در این کامپایلرها، برای الحاق فایل کتابخانه به برنامه باید از دستور زیر به جای دو خط نخست استفاده کنید:

```
#include <iostream.h>
```

از آنجا که خطوط اول و دوم در همهٔ برنامه‌ها تکرار می‌شوند، در فصل‌های آتی، از آوردن آنها چشم‌پوشی شده است. هرچند، باید آنها را به ابتدای بیشتر برنامه‌ها اضافه کنید.

هر برنامهٔ ++C حاوی توابع گوناگونی است؛ اما اجرای دستورات ++C، همیشه از تابع ()main شروع می‌شود. نمادهای { } محدودهٔ شروع و پایان بلوک اصلی برنامه (تابع main) را نشان می‌دهد[۱]. خط نخست این بلوک، جملهٔ معروف "Hello World!" در ++C را در صفحه نمایش (مانیتور) چاپ می‌کند. در زبان ++C، چاپ اطلاعات در صفحهٔ مانیتور با دستور cout انجام می‌شود. خط دوم نیز یک جملهٔ توضیحی است که توسط کامپایلر چشم‌پوشی می‌شود و بنابراین، تاثیری در روند اجرای برنامه ندارد. نمادهای // در آغاز خط، نشانهٔ توضیحی بودن این جمله است. خط آخر نیز مقدار 0 را به سیستم عامل ارسال می‌کند تا اجرای بدون خطای برنامه را اعلام کند. اگر خطایی در برنامه اتفاق بیفتد، عددی غیر از صفر (شمارهٔ خطا) به سیستم عامل ارسال می‌شود.

اگر دستور using namespace std را از ابتدای برنامه حذف کنید، باید دستورات cout را به صورت std::cout بنویسید.

با این وجود، برنامه‌های ++C می‌توانند حاوی بخش‌های متنوع دیگری نیز باشند. ساختار کلی برنامه‌های ++C را می‌توان به صورت زیر بیان کرد:

```
Preprocessor Commands (دستورات پیش‌پردازنده)
Type Definitions (تعریف انواع جدید)
Function Prototypes (عنوان توابع)
Variables (متغیرها)
Functions (main function necessary) (تعریف توابع)
```

توضیحات لازم در مورد پیاده‌سازی هر کدام از این بخش‌ها در فصل‌های آتی آورده شده است.

۱.۲.۱. *استفاده از توضیحات در برنامه*

معمولا برنامه‌های نوشته شده، در آینده به وسیلهٔ برنامه‌نویسان مورد ویرایش قرار می‌گیرند. برای آن که ویرایش برنامه به سهولت انجام شود، لازم است توضیحات لازم در مورد الگوریتم‌ها، دستورات پیچیده و محاسبات ریاضی در درون کد منبع (Source Code) درج شود. یادآوری می‌کنیم که قسمت‌های توضیح در برنامه، توسط کامپایلر نادیده گرفته می‌شود. برای درج توضیحات در برنامه، از روش‌های زیر استفاده می‌شود:

[۱] یک بلوک، مجموعه‌ای از دستورات مرتبط با یکدیگر است که هدف مشخصی را دنبال می‌کند.

استفاده از نمادهای //: این نماد نشانگر توضیح یک خطی است. هر متنی که پس از این نمادها قرار گیرد تا آخر خط به عنوان توضیح در نظر گرفته شده و توسط کامپایلر ترجمه نمی‌شود.

👁 برنامهٔ زیر، نمونه‌ای از کاربرد توضیحات را نشان می‌دهد. (💾 Chapter 1\comment.cpp)

```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello world";  //Program displays a message on-screen
   return 0;
}
```

```
Output:

Hello world
```

استفاده از نمادهای /* ... */: اگر توضیح شما بیش از یک خط داشته باشد بهتر است آن را در بین این نمادها قرار دهید. همچنین، با این روش می‌توانید بخشی از یک خط را به توضیح تبدیل کنید.

👁 در اینجا، برنامهٔ پیشین را با توضیح چند خطی آورده‌ایم. (💾 Chapter 1\multilineComment.cpp)

```
#include <iostream>

using namespace std;

int main()
{
   cout << "hello World";
   /*
   Program displays a
   message on-screen
   */
   return 0;
}
```

از آنجا که درج توضیحات، در خروجی برنامه تاثیری ندارد، خروجی این برنامه مانند مثال پیشین است.

یکی از کاربردهای مهم درج توضیح، غیرفعال کردن موقت کدها است. تبدیل کدها به توضیح، دلایل گوناگونی دارد. برای نمونه، هنگامی که کدهای شما دارای خطا است و زمان کافی برای رفع ایراد آن ندارید می‌توانید کدها را تبدیل به توضیح نمایید. همچنین، جانشین کردن یک سری کد با کدهای جدید می‌تواند دلیلی برای این کار باشد. مشاهدهٔ تاثیر یک کد بر روی برنامه، دلیل دیگری برای تبدیل کدها به توضیح است.

۱.۲.۲. *مفهوم عبارت*

هر برنامه، از یک مجموعه دستورات تشکیل شده است که به هر کدام از آنها یک عبارت (Statement) گفته می‌شود. در واقع، یک عبارت، کوچک‌ترین واحد تشکیل دهندهٔ برنامه‌های C++ است. در زبان محاوره، هر عبارت را می‌توان به یک جمله تشبیه کرد. همان گونه که ما برای رساندن منظور خود، مجموعه‌ای از جمله‌ها را در کنار یکدیگر قرار

می‌دهیم، برای نوشتن یک برنامه، باید عبارت‌های مناسب را در کنار هم قرار دهیم تا هدف مورد نظر به‌دست آید. در ++C، پایان هر عبارت به یک نماد ";" ختم می‌شود. در زیر، چند عبارت ساده را مشاهده می‌کنید:

```
int x;
x = 5;
cout << x;
```

یک عبارت می‌تواند تعریف متغیر، نسبت دادن یک عدد به متغیر یا هر دستور دیگر باشد.

۱.۲.۳. *شناسه‌ها*

شناسه‌ها، اسامی هستند که برای نامگذاری متغیرها، توابع، کلاس‌ها، ماژول‌ها و ... مورد استفاده قرار می‌گیرند. شناسه‌ها با یکی از حروف انگلیسی یا کاراکتر "_" شروع شده و با تعدادی حروف انگلیسی، کاراکتر "_" یا عدد ادامه می‌یابند. استفاده از نمادهایی مانند $، % و # و کاراکترهای فضای خالی (Space و Tab) غیرمجاز هستند. بنابراین، شناسه‌های زیر در ++C معتبر هستند:

x fact _foo mark x1 x_1

اما شناسه‌های زیر نامعتبر بوده و استفاده از آنها تولید خطا خواهد کرد:

1x x# $mark

توجه داشته باشید که شناسه‌ها به بزرگی و کوچکی حروف حساس هستند. بنابراین، شناسه‌هایی مانند Fact و fact از نظر کامپایلر ++C متفاوت هستند.

برای کسب توضیحات بیشتر می‌توانید به فصل "متغیرها، ثابت‌ها و انواع داده" مراجعه کنید.

۱.۳. *دستورات اولیهٔ ورودی و خروجی*

برای تعامل با کاربر (گرفتن داده‌ها از کاربر و نمایش نتیجهٔ پردازش آنها)، لازم است آشنایی کافی با ورودی و خروجی‌ها داشته باشید. در اینجا، با دستورات ورودی و خروجی در C و ++C آشنا خواهید شد.

۱.۳.۱. *استفاده از دستورات cin و cout*

در ++C از دستور cin می‌توان برای ورود داده‌ها از صفحه کلید و از دستور cout برای نمایش داده‌ها در مانیتور استفاده کرد. یادآوری می‌شود که این دستورات به کتابخانهٔ iostream نیاز دارند:

```
cin >> variableName;
cout << variableName or constant;
```

در اینجا، variableName نام متغیر و constant یک مقدار ثابت را مشخص می‌کند.

👁 در نمونهٔ زیر، از دستورات ورودی و خروجی برای دریافت یک عدد صحیح از کاربر و چاپ آن استفاده شده است. (🖫 Chapter 1\io.cpp)

```
#include <iostream>

using namespace std;

int main()
{
   int x;
   cout << "Enter x: ";
   cin >> x;
   cout << "Your input is: " << x;

   return 0;
}
```

```
Output:

Enter x: 5
Your input is: 5
```

برای قالب‌بندی خروجی، از یک سری تغییر دهنده‌ها استفاده می‌شود که در جدول ۱-۱ آورده شده است.

نوع ورودی		نوع ورودی	
dec	نمایش اعداد در مبنای ۱۰	showpos	نمایش نماد + پشت اعداد مثبت
hex	نمایش اعداد در مبنای ۱۶	showbase	نمایش مبنای عدد
oct	نمایش اعداد در مبنای ۸	showpoint	نمایش نقطهٔ اعشار
left	چینش از چپ	uppercase	نمایش اعداد مبنای ۱۶ با حروف بزرگ
right	چینش از راست	noshowpos	نمایش ندادن نماد +
scientific	نمایش اعداد با نماد علمی	noshowbase	نمایش ندادن مبنای عدد

جدول ۱-۱: تغییر دهنده‌های دستور cout

در برنامهٔ زیر، نحوهٔ استفاده از کاراکترهای قالب‌بندی در دستور cout نشان داده شده است. (Chapter 1\format.cpp)

```
#include <iostream>

using namespace std;

int main()
{
   int x;
   x = 3;
   cout.width(10);
   cout << right << "Hello\n";
   cout << showpos << x;

   return 0;
}
```

```
Output:

     Hello
+3
```

در این برنامه، از کاراکتر کنترلی "\n" برای رفتن به ابتدای خط بعدی استفاده شده است. لیست کامل کاراکترهای کنترلی در جدول ۴-۱ آورده شده است.

با افزودن کتابخانهٔ <iomanip> به برنامه می‌توانید از تغییر دهنده‌های دیگری که در جدول ۲-۱ آورده شده است، استفاده کنید.

نوع ورودی		نوع ورودی	
مشخص کردن قالب خروجی	setiosflags	اندازهٔ فضای خروجی	setw
بازنشانی مقادیر پیش‌فرض	resetiosflags	پر کردن فضای خالی با یک کاراکتر	setfill
تعیین دقت اعشاری	setprecision	تعیین مبنای عدد خروجی	setbase

جدول ۲-۱: تغییر دهنده‌های دستور cout

برنامهٔ زیر، نحوهٔ استفاده از تغییر دهنده‌های جدول بالا را نشان می‌دهد. (Chapter 1\iomanip.cpp)

```
#include <iostream>

using namespace std;

int main()
{
   cout << setw(10) << 12 << endl;
   cout << setfill('.') << setw(10) << 12 << endl;
   cout << setbase(16) <<  12 << endl;
   cout << setiosflags(ios::showbase | ios::uppercase) << hex << 12 << endl;
   cout << resetiosflags(ios::uppercase) << 12 << endl;

   return 0;
}
```

```
Output:

        12
........12
c
0XC
0xc
```

دستورات cin و cout در زبان C[1] استاندارد در دسترس نیستند؛ بنابراین، در این زبان باید از دستورات زیر استفاده کرد. این دستورات هم در C استاندارد و هم در C++ قابل استفاده هستند.

```
scanf(const char* format, variables);
printf(const char* format, variables);
```

این دستورات در کتابخانهٔ stdio.h قرار دارند. در ادامه، به بررسی هر کدام از این دستورات خواهیم پرداخت.

[1] استفاده از C استاندارد، هنوز هم برای برنامه‌نویسی انواع بردها کاربرد دارد. بنابراین، یادگیری آن می‌تواند برای دانشجویان (به ویژه دانشجویان رشته‌های سخت‌افزار و مخابرات) مفید باشد.

۱.۳.۲. دستور scanf()

بخش format در تابع scanf()، یک رشته[۱] است که نوع داده‌های ورودی را مشخص می‌کند. هر نوع داده، با یک حرف متصل به نماد % نشان داده می‌شود. انواع داده‌ها در جدول ۳-۱ نشان داده شده است. این داده‌ها در فصل دوم توضیح داده شده‌اند.

%	نوع ورودی	%	نوع ورودی
c	کاراکتر	s	رشته
d	عدد دسیمال	i	عدد (integer)
u	عدد دسیمال مثبت	o	عدد مثبت در مبنای ۸
n	تعداد کاراکترهای ورودی[۲]	x	عدد مثبت در مبنای ۱۶
f	عدد اعشاری	e	عدد اعشاری
p	اشاره‌گر	g	عدد اعشاری

جدول ۳-۱: کاراکترهای قالب‌بندی دستور scanf()

نمونهٔ زیر، استفاده از این توابع را نشان می‌دهد: (Chapter 1\scanf.cpp)

```
#include <stdio.h>

void main()
{
   printf("Enter a character: ");
   char ch;
   scanf("%c", &ch);
   printf("%c = %d", ch, ch);
}
```

```
Output:

Enter a character: A
A = 65
```

توجه کنید که در زبان C استاندارد نمی‌توان از دستور using namespace استفاده کرد.

۱.۳.۳. دستور printf()

این دستور، برای نوشتن یک متن در مانیتور مورد استفاده قرار می‌گیرد. مقدار برگشتی این تابع، تعداد حروف چاپ شده است. بخش format دستور printf()، نحوهٔ نمایش اطلاعات در خروجی را نشان می‌دهد. این بخش شامل اطلاعات زیر است:

---

[۱] مجموعه‌ای از کاراکترهای کنار هم را یک رشته می‌نامند.

[۲] یک ورودی نیست.

- متنی که باید عینا چاپ شود.
- کاراکترهای کنترلی که شکل خروجی را مشخص می‌کند. این کاراکترها در جدول ۴-۱ آورده شده‌اند.
- کاراکترهای تعیین کنندهٔ نوع خروجی که با نماد % مشخص می‌شوند. لیست این کاراکترها در جـدول ۵-۱ آورده شده‌اند.

کاراکتر کنترلی	مفهوم	کاراکتر کنترلی	مفهوم
\n	ایجاد خط جدید (Newline)	\"	چاپ کاراکتر " (Double Quote)
\f	کاراکتر Form Feed	\'	چاپ کاراکتر ' (Single Quote)
\r	کاراکتر Carriage Return	\t	معادل کلید Tab
\a	صدای بوق (Beep)	\v	تب عمودی (Vertical Tab)
\xnnn	یک کاراکتر در مبنای شانزده	\?	کاراکتر نماد سوال
\nnn	یک کاراکتر در مبنای هشت	\\	چاپ نماد \ (Backslash)
\b	کاراکتر Backspace		

جدول ۴-۱: کاراکترهای کنترلی

%	مفهوم	%	مفهوم
c	کاراکتر	x	عدد مثبت در مبنای ۱۶ با حروف کوچک
d	عدد دسیمال (دهدهی)	X	عدد مثبت در مبنای ۱۶ با حروف بزرگ
u	عدد دسیمال مثبت	s	رشته
E یا e	نمایش عدد با نماد علمی	i	عدد (integer)
f	نمایش عدد اعشاری	p	اشاره‌گر
G یا g	نمایش عدد اعشاری	n	شمارش تعداد کاراکترهای چاپ شده
o	عدد مثبت در مبنای ۸	%	نمایش نماد %

جدول ۵-۱: کاراکترهای قالب‌بندی دستور printf()

با کمک کاراکتر * (ستاره) شکل زیر را ترسیم کنید. (Chapter 1\stars.cpp)

```
*
**
***
****
*****
```

برای ترسیم این شکل، از برنامهٔ زیر استفاده می‌کنیم:

```
#include <stdio.h>

int main()
{
   printf("*\n**\n***\n****\n*****\n");
```

```
    return 0;
}
```

همان گونه که می‌بینید، در اینجا از کاراکتر '\n' برای تفکیک سطرها استفاده شده است.

👁 برنامه‌ای بنویسید که نام و سن شما را در دو سطر جداگانه نمایش دهد. (💾 Chapter 1\printf.cpp)

```
#include <stdio.h>

int main()
{
    int age = 20;
    printf("Name: Mohammad\n");
    printf("Age: %d", age);

    return 0;
}
```

```
Output:

Name: Mohammad
Age: 20
```

۱.۳.۴. *گرفتن یک کاراکتر از کاربر*

اگر قصد دارید تنها یک کاراکتر از کاربر دریافت کنید، به جای دستورات ()scanf یا cin از یکی از دستورات ()getch یا ()getche استفاده نمایید. این توابع در کتابخانهٔ conio.h قرار دارند. نحوهٔ استفاده از این دستورات به صورت زیر است:

```
variableName = getch();
```

یا

```
variableName = getche();
```

تفاوت این دستورات، تنها در نمایش کاراکتر در هنگام گرفتن کاراکتر است. دستور ()getch کاراکتر دریافت شده را نمایش نمی‌دهد؛ اما دستور ()getche یک کاراکتر را دریافت کرده و سپس آن را در مانیتور نمایش می‌دهد.

👁 برنامهٔ زیر، نحوهٔ استفاده از دستور ()getche را نشان می‌دهد. (💾 Chapter 1\getche.cpp)

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

using namespace std;

int main()
{
    char ch;
    cout << "Press any key: ";
    ch = getche();
    cout << "\nA key is pressed from keyboard: " << ch;

    return 0;
}
```

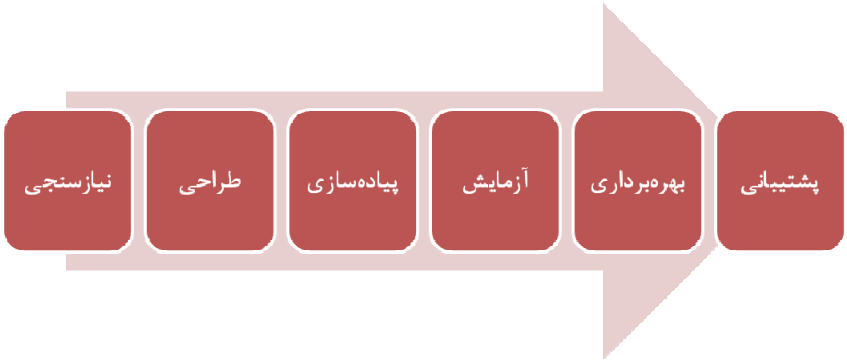
```
Output:

Press any key: A
A key is pressed from keyboard: A
```

این برنامه، یک کاراکتر را از کاربر گرفته و سپس آن را در خروجی چاپ می‌کند.

۱.۴. مراحل نوشتن یک برنامه

نوشتن یک برنامه، مانند حل یک مسالهٔ ریاضی است. برای رسیدن به جواب یک مساله، باید راه حل ارائه کرده و مراحلی را طی نمایید. برای حل یک مساله ممکن است راه حل‌های گوناگونی وجود داشته باشد.



برای نوشتن یک برنامهٔ استاندارد، باید مراحل زیر را دنبال کنید:

۱. تعیین نیازمندی‌های مساله: در این مرحله، باید نیازهای سخت‌افزاری و نرم‌افزاری مساله را تعیین کنید.
۲. طراحی الگوریتم: در این گام، مراحل انجام کار با توجه به شرایط موجود مشخص می‌شوند. دقت کنید که هر مساله می‌تواند راه حل‌های گوناگونی داشته باشد؛ اما، برنامه‌نویس حرفه‌ای کسی است که بهترین راه حل را از میان آنها انتخاب کند. بسیاری از برنامه‌ها، دارای سه بخش دریافت داده، پردازش و خروجی (چاپ اطلاعات) هستند.

۳. پیاده‌سازی: در این مرحله، برنامه‌ها با یک زبان برنامه‌نویسی مانند C++ نوشته می‌شوند.
۴. تست برنامه: تست برنامه، یکی از سخت‌ترین مراحل تولید نرم‌افزار است. برنامهٔ تولید شده، باید با داده‌های گوناگون تست شده و استثناهای برنامه مشخص شود.
۵. بهره‌برداری: در این مرحله، برنامه در اختیار کاربران قرار می‌گیرد.
۶. پشتیبانی: پس از آن که محصول نهایی (برنامهٔ اجرایی) وارد بازار شد؛ لازم است خطاهای منطقی به وجود آمده که تاکنون با آنها برخورد نکرده بودید تصحیح شوند. همچنین، لازم است نیازهای روز کاربران را به آن اضافه کرده و برنامهٔ خود را به‌روزرسانی کنید. معمولا، برای برنامه‌ها، یک دورهٔ پشتیبانی در نظر گرفته می‌شود.

۱.۵. خطایابی در برنامه

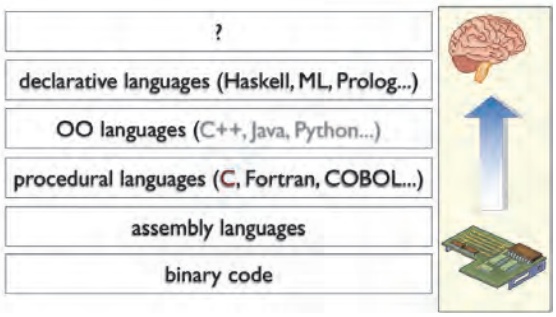
در هنگام برنامه‌سازی ممکن است خطاهای گوناگونی در برنامه رخ دهد. دوری از این خطاها گریزناپذیر است؛ اما نکتهٔ مهم این است که برنامه‌نویس بتواند آنها را کشف و به درستی رفع کند.

خطاهایی که در هنگام برنامه‌نویسی رخ می‌دهند، به دو دستهٔ خطاهای گرامری (Syntax) و خطاهای منطقی (Logic) تقسیم می‌شوند. خطاهای گرامری، غلط‌های املایی هستند که در هنگام تایپ برنامه رخ داده و به راحتی قابل شناسایی و رفع هستند. برای نمونه، اگر نماد ";" را در پایان یک دستور قرار ندهید، خطای گرامری صادر شده و در نتیجه، برنامه کامپایل نمی‌شود. دستهٔ دوم، خطاهای منطقی هستند که در آنها، منطق برنامه دچار مشکل است. در این گونه برنامه‌ها، اگرچه برنامه اجرا می‌شود، اما نتیجهٔ اجرای آن درست نیست. برای حل خطاهای منطقی لازم است تست‌های گوناگونی روی برنامه انجام شود. در بسیاری از مواقع، خطاهای منطقی توسط کامپایلر اعلام نمی‌شود؛ مگر آن که خطا، اجرای برنامه را مختل نماید. کشف این نوع خطاها ساده نبوده و در بسیاری از مواقع، وقت‌گیر است.

۱.۶. چکیدهٔ فصل

مجموعه‌ای از دستورات کامپیوتری که در کنار هم کار خاصی را انجام می‌دهند، برنامه نامیده می‌شود. به کسی که با چیدن دستورات در کنار هم برنامهٔ کامپیوتری (نرم‌افزار) تولید می‌کند برنامه‌نویس می‌گویند. زبان C++ یکی از زبان‌های بسیار مفید در تولید برنامه‌ها است. این زبان، زبانی ساخت‌یافته و شئ‌گرا است که امکان نوشتن، مدیریت و پشتیبانی برنامه‌های بزرگ را فراهم می‌آورد.

زبان‌های برنامه‌نویسی، به انواع سطوح پایین، میانی و بالا تقسیم می‌شوند. زبان‌های سطح بالا، زبان‌هایی هستند که به زبان انسان نزدیک‌ترند؛ اما زبانی که کاملا شبیه زبان‌های محاوره‌ای باشد وجود ندارد. زبان‌های سطح پایین هم به زبان سخت‌افزار نزدیک هستند. اگر چه زبان‌های سطح پایین کدهای بهینه‌تری تولید می‌کنند؛ اما نوشتن برنامه با این زبان‌ها، بسیار زمان‌بر و طاقت‌فرسا است. زبان C++ یک زبان میانی است که امکان برنامه‌نویسی سخت‌افزاری و کاربردی را به طور همزمان مهیا می‌کند. شکل زیر، سطح نسبی انواع زبان‌های برنامه‌نویسی را نشان می‌دهد.



تبدیل یک برنامهٔ سطح بالا به زبان ماشین را کامپایل (Compile) می‌گویند. نوشتن برنامه‌های C++ در کامپایلرهایی مانند Dev CPP، Free C++، Turbo C++ و CodeBlocks امکان‌پذیر است. همچنین برای کامپایل و اجرای برنامه‌ها می‌توانید از کامپایلرهای آنلاین در اینترنت استفاده کنید.

در زبان C استاندارد، برای ورود اطلاعات از صفحه کلید، از دستور ()scanf استفاده می‌شود. در ++C به جای این تابع می‌توانید از دستور cin استفاده کنید که قابلیت‌های بیشتری در اختیار شما قرار می‌دهد. همچنین، در زبان C برای چاپ اطلاعات در صفحه نمایش از دستور ()printf استفاده می‌شود که معادل آن در ++C دستور cout است. برای ورود یک کاراکتر از صفحه کلید هم از دستور ()getch یا ()getche استفاده می‌شود.

به راه حلی که برای حل یک مساله داده می‌شود، الگوریتم می‌گویند. بیشتر برنامه‌های رایانه‌ای، در سه مرحله نوشته می‌شوند: دریافت داده‌ها، مرحله پردازش و ارسال اطلاعات به خروجی[۱].

تمرین:

۱. برنامه‌ای بنویسید که طول و عرض یک مستطیل را گرفته و مساحت آن را حساب کند.
۲. برنامه‌ای بنویسید که نام و سن شما را در دو سطر جداگانه چاپ کند.
۳. آیا برنامهٔ زیر به درستی کار می‌کند؟

```
int main()
{
        int
         i;
        for
      (i = 0;
     i <10; i =
    i + 1) printf
  ("%d\n",     i);
 return             0;
}
```

[۱] به داده‌های پردازش شده، اطلاعات گفته می‌شود.