

# فصل دوم

---

## مقدمه‌ای بر برنامه‌نویسی C++

---

### اهداف

- نوشتن برنامه‌های ساده کامپیوتری در C++.
- نوشتن عبارات ساده ورودی و خروجی.
- استفاده از نوع‌های بنیادین.
- مفاهیم حافظه.
- استفاده از عملگرهای محاسباتی.
- نوشتن عبارات ساده تصمیم‌گیری.



رئوس مطالب
۲-۱ مقدمه
۲-۲ یک برنامه ساده: چاپ یک عبارت متنی
۲-۳ اصلاح برنامه
۲-۴ یک برنامه ساده دیگر: جمع اعداد صحیح
۲-۵ مفاهیم حافظه
۲-۶ محاسبات
۲-۷ تصمیم‌گیری: عملگرهای مقایسه‌ای و رابطه‌ای
۲-۸ مبحث آموزشی مهندسی نرم‌افزار: بررسی نیازمندی‌های ATM

### ۲-۱ مقدمه

در این فصل به معرفی برنامه‌نویسی C++ می‌پردازیم، که طراحی برنامه‌ها را تسهیل خواهد بخشید. اکثر برنامه‌های که در این کتاب با آنها مواجه خواهید شد، مبادرت به پردازش اطلاعات و نمایش نتایج می‌کنند. در این فصل، به معرفی پنج مثال می‌پردازیم که نحوه نمایش پیغام و همچنین دریافت داده از کاربران را به شما نشان می‌دهند. سه مثال اول، ساده بوده و تنها مبادرت به نمایش پیغام در صفحه نمایش می‌کنند. برنامه بعدی، دو عدد از کاربر دریافت کرده و مجموع آنها را محاسبه و نتیجه را بنمایش درمی‌آورد. در ضمیمه بحث، شما را با نحوه انجام انواع محاسبات و ذخیره نتایج برای استفاده‌های بعدی آشنا خواهیم کرد. مثال پنجم در ارتباط با تصمیم‌سازی است که مبادرت به مقایسه دو عدد کرده و سپس پیغامی براساس مقایسه بنمایش درمی‌آورد. برای اینکه درک مناسب و آسانی از برنامه‌نویسی C++ بدست آورید، خط به خط هر برنامه را تحلیل می‌کنیم. برای اینکه مهارت‌های کسب کرده خود از این فصل را بکار گیرید، چند مسئله برنامه‌نویسی در بخش تمرینات در نظر گرفته‌ایم.

### ۲-۲ یک برنامه ساده: چاپ یک عبارت متنی

زبان C++ از نمادهای استفاده می‌کند که ممکن است برای غیر برنامه‌نویسان عجیب و غریب بنظر برسند. در این بخش به بررسی برنامه‌ای می‌پردازیم که عبارتی را در یک خط چاپ می‌کند. برنامه و خروجی آن در شکل ۲-۱ آورده شده است. این برنامه حاوی چندین ویژگی مهم زبان C++ است. برای آشنایی بهتر شما، به توضیح خط به خط برنامه می‌پردازیم.

در خطوط 1 و 2

```
// Fig. 2.1: fig02_01.cpp  
// Text-printing program.
```



### مقدمه ای بر برنامه‌نویسی C++ \_\_\_\_\_ فصل دوم 3

که با //، آغاز شده‌اند، نشان‌دهنده این مطلب هستند که این خطوط توضیح می‌باشند. برنامه‌نویسان توضیحات را در برنامه یا لیست کد قرار می‌دهند تا خوانایی کدهای خود را افزایش دهند. توضیحات می‌توانند در خط متعلق به خود جای داده شوند که آنها را توضیحات تمام خط می‌نامیم یا در انتهای یک خط از کد که آنها را توضیحات انتهای خط می‌نامیم، قرار بگیرند. کامپایلر C++ توضیحات را نادیده می‌گیرد، به این معنی که توضیحات هیچ تأثیری در اجرای برنامه ندارند. توضیح بکار رفته در خط 1 فقط نشان‌دهنده شماره تصویر و نام فایل این برنامه است. البته برخی از برنامه‌نویسان عبارات توضیحی خود را مابین کاراکترهای /\* و \*/ قرار می‌دهند.

```
1 // Fig. 2.1: fig02_01.cpp
2 // Text-printing program.
3 #include <iostream> // allows program to output data to the screen
4
5 // function main begins program execution
6 int main()
7 {
8     std::cout << "Welcome to C++!\n"; // display message
9
10    return 0; // indicate that program ended successfully
11
12 } // end function main
```

Welcome to C++!

شکل ۱-۲ | برنامه چاپ عبارت.

برنامه‌نویسی ایده‌آل



هر برنامه‌ای باید با یک یا چندین توضیح در ارتباط با اهداف برنامه و همچنین نام برنامه‌نویس، تاریخ و زمان آغاز

شود.

خط 3

```
#include <iostream> // allows program to output data to the screen
```

یک رهنمود پیش‌پردازنده است، که پیغامی برای پیش‌پردازنده C++ می‌باشد. خطوطی که با نماد # آغاز می‌شوند، قبل از اینکه برنامه کامپایل شود توسط پیش‌پردازنده پردازش می‌شوند. این خط به پیش‌پردازنده اعلان می‌کند تا محتویات سرآیند جریان ورودی/خروجی فایل `<iostream>` را در برنامه وارد سازد. بایستی این فایل در هر برنامه‌ای که می‌خواهد داده‌ای به صفحه نمایش انتقال دهد، یا اینکه از صفحه کلید و با استفاده از جریان ورودی/خروجی C++ داده‌ای دریافت نماید، وارد شود. در شکل ۱-۲ خروجی برنامه آورده شده است. در فصل ششم با فایل‌های سرآیند در فصل پانزدهم با محتویات `iostream` آشنا خواهید شد.



## 4 فصل دوم \_\_\_\_\_ مقدمه‌ای بر برنامه‌نویسی C++

### خطای برنامه‌نویسی



در صورتیکه افزودن فایل سرآیند `<iostream>` به برنامه‌ای که میادرت به دریافت داده از صفحه کلید یا ارسال داده به صفحه نمایش می‌کند، فراموش شود، کامپایلر یک پیغام خطا ارسال خواهد کرد.

خط 4 فقط یک خط خالی است. برنامه‌نویسان برای اینکه قرائت‌های خود را آسانتر کنند از خطوط خالی، کاراکترهای فاصله (space) و تب (tab) استفاده می‌کنند. به مجموعه این کاراکترها، white space می‌گویند. معمولاً این کاراکترها توسط کامپایلر نادیده گرفته می‌شوند. در این فصل و چند فصل بعدی، در ارتباط با قواعد رایج در استفاده از کاراکترهای white-space صحبت خواهیم کرد تا خوانایی برنامه‌ها افزایش یابد.

### برنامه‌نویسی ایده‌آل



با استفاده از خطوط خالی و کاراکترهای فاصله، خوانایی برنامه‌ها را افزایش دهید.

خط 5

```
// function main begins program execution
```

هم یک خط توضیحی تمام خط است که اعلان می‌کند برنامه از خط بعدی شروع می‌شود.

خط 6

```
int main()
```

بخشی از هر برنامه C++ است. پرانتزهای واقع پس از `main` نشان می‌دهند که `main` یک بلوک برنامه بنام تابع است. برنامه‌های C++ می‌توانند حاوی یک یا چندین تابع و کلاس باشند، اما بایستی یکی از آنها حتماً `main` باشد، حتی اگر `main` اولین تابع در برنامه نباشد. کلمه کلیدی `int` که در سمت چپ `main` قرار گرفته، بر این نکته دلالت دارد که `main` یک مقدار صحیح "برمی‌گرداند" (عدد بدون اعشار). کلمه کلیدی، کلمه‌ای در کد است که توسط C++ رزرو شده است. لیست کامل کلمات کلیدی C++ در جدول شکل ۳-۴ آورده شده‌اند. به هنگام مطالعه فصل سوم و ششم به توضیح مفهومی دقیق "مقدار برگشتی" از سوی یک تابع خواهیم پرداخت. اما برای این لحظه، کافیسیت بدانید که کلمه کلیدی `int` در سمت چپ برنامه‌های شما قرار خواهد گرفت.

بایستی براکت چپ، {، (خط 7) در ابتدای بدنه هر روالی قرار داده شود. براکت متناظر، براکت

راست، }، (خط 12) است، که باید آنرا در انتهای بدنه هر روالی قرار داد. خط 8

```
std::cout << "Welcome to C++!\n"; // display message
```



## مقدمه ای بر برنامه‌نویسی C++ \_\_\_\_\_ فصل دوم 5

به کامپیوتر فرمان می‌دهد تا رشته‌ای از کاراکترها را که مابین جفت کوتیشن قرار دارند بر روی صفحه نمایش چاپ کند. گاهی اوقات رشته بنام رشته کاراکتری، پیغام یا رشته لیترال هم خوانده می‌شود. کاراکترهای *white space* توسط کامپایلر نادیده گرفته می‌شوند.

کل خط 8، شامل `std::cout` عملگر <<، رشته `"\n ! Welcome to C++"` و یک سیمیکولن (;) است، که به کل این خط یک عبارت گفته می‌شود. هر عبارتی در C++ باید با یک سیمیکولن خاتمه پذیرد (که به آن *خاتمه‌دهنده عبارت* هم گفته می‌شود). رهنمودهای پیش‌پردازنده (همانند `#include`) با سیمیکولن خاتمه نمی‌یابند. خروجی و ورودی در C++ با *جریانی* (*stream*) از کاراکترها پیاده‌سازی می‌شود. بنابر این، زمانیکه عبارت قبلی اجرا می‌شود، مبادرت به ارسال جریانی از کاراکترهای *Welcome* **to C++** ! به شی *جریان خروجی استاندارد* (`std::cout`) که معمولاً با صفحه نمایش مرتبط است می‌کند. در فصل پانزدهم به بررسی بیشتر `std::cout` خواهیم پرداخت.

دقت کنید که `std::` قبل از `cout` قرار گرفته است. انجام این عمل به هنگام استفاده از رهنمود پیش‌پردازنده `<iostream> #include` الزامی است. نماد `std::cout` نشان می‌دهد که در حال استفاده از یک نام هستیم، که این نام در این مورد `cout` می‌باشد، که متعلق به "فضای نامی" `std` است. فضاهای نامی از ویژگیهای پیشرفته C++ هستند. در فصل بیست و چهارم بطور کامل با فضاهای نامی آشنا خواهید شد. اما برای این لحظه، باید بخاطر داشته باشید که کلمه `std::` را قبل از هر کدامیک از نمادهای `cout`، `cin` و `cerr` در یک برنامه قرار دهید. در برنامه شکل ۱۳-۲ از این نمادها به همراه عبارت `using` استفاده شده است، که ما را قادر به حذف `std::` قبل از هر استفاده از فضای نامی `std` می‌کند.

عملگر << نشان‌دهنده، عملگر *درج جریان* است. هنگامی که این برنامه اجرا می‌شود، مقدار موجود در سمت راست این عملگر، *عملوند* سمت راست، وارد جریان خروجی می‌گردد. دقت کنید که عملگر مستقیماً به مکانی اشاره می‌کند که داده باید به آنجا برود. معمولاً کاراکترهای قرار گرفته در سمت راست عملگر به همان شکلی که مابین جفت کوتیشن‌ها آورده شده‌اند، چاپ می‌شوند. با این وجود، توجه نمائید که کاراکتر `\n` بر روی صفحه نمایش چاپ نمی‌شود. کاراکتر `\` کاراکتر *escape* نامیده می‌شود. این کاراکتر نشان می‌دهد که یک کاراکتر ویژه چاپ خواهد شد. زمانیکه در دنباله‌ای از رشته‌های کاراکتری یک کاراکتر `\` وارد شود، کاراکتر پس از آن بعنوان یک *توالی escape* در نظر گرفته خواهد شد. در این برنامه توالی `escape` کاراکتر `\n` است، که به مفهوم *خط جدید* یا *newline* می‌باشد. این کاراکتر سبب



## 6 فصل دوم \_\_\_\_\_ مقدمه‌ای بر برنامه‌نویسی C++

می‌شود تا کرسر به ابتدای خط بعدی در صفحه نمایش منتقل شود. در جدول شکل ۲-۲ تعدادی از توالی‌های escape که از کاربرد بیشتری برخوردار هستند آورده شده‌اند.

توضیحات	escape توالی
خط جدید. کرسر را در ابتدای خط بعدی قرار می‌دهد.	\n
تب افقی. کرسر را به اندازه یک تب (tab) به جلو انتقال می‌دهد.	\t
کرسر را به ابتدای خط جاری باز می‌گرداند. آنرا با \n اشتباه نگیرید.	\r
هشدار. زنگ یا صدای سیستم را فعال می‌کند.	\a
کاراکتر \. برای چاپ کاراکتر \ استفاده می‌شود.	\\
کاراکتر '. برای چاپ کاراکتر ' استفاده می‌شود.	'
جفت کوتیشن. از این توالی برای چاپ کاراکتر جفت کوتیشن استفاده می‌شود.	"

شکل ۲-۲ توالی‌های escape.

### خطای برنامه‌نویسی



فراموش کردن سیمیکولن در پایان یک عبارت، خطای نحوی (syntax error) بدنبال خواهد داشت. این نوع خطا زمانی رخ می‌دهد که کامپایلر با عبارتی مواجه شود و نتواند مفهوم آنرا تشخیص دهد. معمولاً در چنین حالتی کامپایلر یک پیغام خطا ارسال می‌کند تا به برنامه‌نویس در یافتن مکان خطا و رفع آن کمک نماید. خطاهای نحوی به هنگام خروج از قواعد نگارش زبان رخ می‌دهند. گاهی به این خطاها، خطاهای کامپایلر هم گفته می‌شود، چراکه این خطاها در زمان فاز کامپایل شدن برنامه رخ می‌دهند.

خط 10

```
return 0; // indicate that program ended successfully
```

در پایان هر تابع main وجود خواهد داشت. کلمه کلیدی return یکی از چندین روش موجود برای خروج از یک تابع است. زمانی که عبارت return در پایان تابع main بکار گرفته می‌شود، همانند این برنامه، مقدار 0 نشان‌دهنده این مطلب است که برنامه با موفقیت به کار خود پایان داده است. در فصل ششم، با جزئیات عملکرد توابع و دلایل افزودن چنین عبارتی به آنها بیشتر آشنا خواهید شد. اما برای این لحظه، کافایت بدانید که این عبارت در انتهای هر برنامه‌ای قرار داده می‌شود. براکت راست، {، (خط 12) انتهای تابع main را نشان می‌دهد.

### برنامه‌نویسی ایده‌آل



بسیاری از برنامه‌نویسان در انتهای آخرین کاراکتر چاپ شده توسط یک تابع، کاراکتر \n را قرار می‌دهند. با این عمل مطمئن می‌شوند که تابع در پایان کار خود، کرسر را در ابتدای خط بعدی قرار خواهد داد. این عمل می‌تواند در راستای ایجاد برنامه‌هایی با قابلیت استفاده مجدد موثر باشد.

### برنامه‌نویسی ایده‌آل





## مقدمه ای بر برنامه‌نویسی C++ فصل دوم 7

به دندان‌دار بودن خطوط کد دقت کنید. این عمل یکی از قواعد بکارگیری فاصله‌ها در برنامه است. دندان‌دار کردن کد برنامه‌ها، باعث افزایش خوانایی و گرامر آن می‌شود.

### برنامه‌نویسی ایده‌آل



برای حفظ ظاهر آراسته و مرتب در میان کدهای نوشته شده، بهتر است میزان دندان‌گذاری مورد نظر خود را از همان ابتدا مشخص سازید. پیشنهاد می‌کنیم تا از کلید `tab` با فاصله‌گذاری  $\frac{1}{4}$  اینچ یا سه فاصله (`space`) از سطح دندان‌گذاری شده فوقانی استفاده کنید.

### ۲-۳ اصلاح برنامه

این بخش ادامه‌دهنده معرفی برنامه‌نویسی C++ با دو مثال است، که نحوه اصلاح برنامه مطرح شده در شکل ۲-۱ را نشان می‌دهند. در مثال اول، متن در یک خط با استفاده از چندین عبارت چاپ می‌شود. در مثال دوم همان متن توسط یک عبارت و در چندین خط چاپ می‌گردد.

#### چاپ متن در یک خط توسط چند عبارت

رشته `Welcome to C++!` می‌تواند به چندین روش چاپ شود. برای مثال، در برنامه شکل ۲-۳ از دو عبارت برای وارد کردن جریان (خطوط ۸-۹) استفاده شده است، تا همان خروجی برنامه شکل ۲-۱ تولید شود. خروجی برنامه دوم دقیقاً همانند برنامه اول است، چرا که هر عبارت از مکانی که عبارت قبلی به عمل چاپ پایان داده، شروع به چاپ می‌کند. عبارت اول کلمه `Welcome` و بدنبال آن یک فاصله چاپ کرده و عبارت دوم هم کار چاپ را از همان خط و پس از فاصله چاپ شده دنبال می‌کند. بطور کلی، C++ به برنامه‌نویس امکان می‌دهد تا عبارات را به روش‌های گوناگون بکار گیرد.

#### چاپ متن در چند خط توسط یک عبارت

همانند برنامه شکل ۲-۴ می‌توان یک عبارت را در چندین خط به نمایش در آورد. هر بار که عبارت ارسالی به خروجی با کاراکتر توالی `\n` مواجه شود، کرسر به ابتدای خط بعدی منتقل خواهد شد. برای ایجاد یک خط خالی در خروجی، دو کاراکتر خط جدید را پشت سرهم، همانند خط ۸ قرار دهید.

```

1 // Fig. 2.3: fig02_03.cpp
2 // Printing a line of text with multiple statements.
3 #include <iostream> // allows program to output data to the screen
4
5 // function main begins program execution
6 int main()
7 {
8     std::cout << "Welcome ";
9     std::cout << "to C++!\n";
10
11     return 0; // indicate that program ended successfully
12
13 } // end function main

```

Welcome to C++!



## 8 فصل دوم \_\_\_\_\_ مقدمه‌ای بر برنامه‌نویسی C++

شکل ۲-۳ | چاپ متن در یک خط با چند عبارت.

```
1 // Fig. 2.4: fig02_04.cpp
2 // Printing multiple lines of text with a single statement.
3 #include <iostream> // allows program to output data to the screen
4
5 // function main begins program execution
6 int main()
7 {
8     std::cout << "Welcome\nto\nnC++!\n";
9
10    return 0; // indicate that program ended successfully
11
12 } // end function main
```

```
Welcome
to

C++!
```

شکل ۲-۴ | چاپ متن در چند خط با یک عبارت.

### ۲-۴ یک برنامه ساده دیگر: جمع اعداد صحیح

در برنامه بعدی از شی جریان ورودی `std::cin` و عملگر استخراج `>>`، برای بدست آوردن دو عدد صحیح تایپ شده از سوی کاربر از طریق صفحه کلید، جمع آنها و نمایش نتیجه بدست آمده با استفاده از `std::cout` می‌پردازیم. برنامه شکل ۲-۵ عمل جمع و خروجی حاصل از این برنامه را نشان می‌دهد.

```
1 // Fig. 2.5: fig02_05.cpp
2 // Addition program that displays the sum of two numbers.
3 #include <iostream> // allows program to perform input and output
4
5 // function main begins program execution
6 int main()
7 {
8     // variable declarations
9     int number1; // first integer to add
10    int number2; // second integer to add
11    int sum; // sum of number1 and number2
12
13    std::cout << "Enter first integer: "; // prompt user for data
14    std::cin >> number1; // read first integer from user into number1
15
16    std::cout << "Enter second integer: "; // prompt user for data
17    std::cin >> number2; // read second integer from user into number2
18
19    sum = number1 + number2; // add the numbers; store result in sum
20
21    std::cout << "Sum is " << sum << std::endl; //display sum; end line
22
23    return 0; // indicate that program ended successfully
24
25 } // end function main
```





## مقدمه ای بر برنامه نویسی C++ فصل دوم 9

```
Enter first integer: 45
Enter second integer: 72
Sum is 117
```

شکل ۲-۵ | برنامه جمع که مجموع دو عدد صحیح دریافتی از صفحه کلید را محاسبه می کند.

توضیحات موجود در خطوط 1 و 2

```
// Fig. 2.5: fig02_05.cpp
// Addition program that displays the sum of two numbers.
```

نشاندنده نام فایل و هدف برنامه هستند. رهنمود پیش پردازنده C++

```
#include <iostream> // allows program to perform input and output
```

در خط 3 حاوی محتویات سرآیند فایل *iostream* در برنامه است.

همانطوری که قبلاً هم گفته شد، هر برنامه ای با اجرای تابع **main** آغاز می شود (خط 6). براکت سمت چپ (خط 7) ابتدا و آغاز بدنه **main** و براکت متناظر سمت راست (خط 25)، انتها و پایان بدنه را نشان می دهد.

خطوط 9-11

```
int number1; // first integer to add
int number2; // second integer to add
int sum; // sum of number1 and number2
```

بخش اعلان می باشند. کلمات **number1**، **number2** و **sum** اسامی متغیرها هستند. متغیرها مکان های در حافظه کامپیوتر هستند که می توان مقادیری را برای استفاده برنامه در آنها ذخیره کرد. این اعلان ها مشخص می کنند که متغیرهای **number1**، **number2** و **sum** از نوع داده **int** هستند، به این معنی که این متغیرها قادر به نگهداری مقادیر صحیح همانند اعداد 0، 31914، -11، 7، و غیره می باشند. تمامی متغیرها بایستی دارای نام و یک نوع داده باشند تا بتوان از آنها در برنامه استفاده کرد. چندین متغیر از یک نوع را می توان در خط یا چند خط مجزا از هم اعلان کرد. می توانیم هر سه متغیر فوق را در یک خط و بصورت زیر اعلان کنیم:

```
int number1, number2, sum;
```

با این همه، در این حالت خوانائی برنامه کاهش یافته و مانع می شود تا توضیحات مناسب برای هر متغیر و هدف از آن متغیر در برنامه را وارد سازیم. اگر بخواهیم بیش از یک متغیر در یک خط اعلان کنیم



## 10 فصل دوم \_\_\_\_\_ مقدمه ای بر برنامه نویسی C++

(همانند مورد بالا) باید اسامی را با یک ویرگول (,) از یکدیگر جدا کنیم. که به این حالت لیست جدا شده با ویرگول گفته می شود.

### برنامه نویسی ایده‌ال



پس از هر ویرگول (,) یک فاصله قرار دهید تا خوانایی برنامه افزایش یابد.

### برنامه نویسی ایده‌ال



برخی از برنامه نویسان ترجیح می دهند تا هر متغیر را در یک خط جداگانه اعلان کنند. در این حالت قرارداد یک توضیح در کنار هر اعلان به آسانی صورت می گیرد.

بزودی در مورد نوع داده *double* (خاص اعداد حقیقی، اعداد با نقطه اعشار همانند 3.4 و 11.19-) و نوع داده *char* (خاص داده کاراکتری، یک متغیر از نوع *char* فقط می تواند یک حرف کوچک، یک حرف بزرگ، یک رقم یا یک کاراکتر خاص همانند \$ یا \* را در خود ذخیره سازد) صحبت خواهیم کرد.

غالباً به نوع های همانند *int*، *double* و *char* نوع های بنیادین، نوع های اصلی یا نوع های توکار (*built-in*) می گویند. انواع نوع های بنیادین از جمله کلمات کلیدی هستند و از اینرو باید تماماً با حروف کوچک به نمایش در آیند.

نام یک متغیر (همانند *number1*) می تواند هر شناسه معتبری که یک کلمه کلیدی نیست، باشد. یک شناسه متشکل از دنباله ای از کاراکترها شامل حروف، ارقام و خط زیرین ( \_ ) است. یک شناسه نمی تواند با یک رقم آغاز شود. زبان C++ از جمله زبان های حساس به موضوع است، به این معنی که مابین حروف کوچک و بزرگ تفاوت قائل می شود. از اینرو شناسه های *a1* و *A1* با هم برابر نیستند.

### قابلیت حمل



زبان C++ به شناسه ها امکان می دهد تا هر طولی داشته باشند، اما امکان دارد سیستم شما یا ساختار C++ بکار رفته محدودیت هایی بر روی طول شناسه ها اعمال کند. از اینرو برای حفظ سازگاری و قابلیت حمل، از شناسه هایی با طول 31 کاراکتر یا کمتر استفاده کنید.

### برنامه نویسی ایده‌ال



انتخاب اسامی با معنی به برنامه کمک می کند که خود به عنوان توضیحی بر برنامه باشد (*self-documenting*). در چنین حالتی اگر متن برنامه در اختیار دیگران قرار داده شود، بدون اینکه نیازی به راهنما و توضیحات اضافی باشد، عملکرد برنامه مشخص خواهد بود.

### برنامه نویسی ایده‌ال



از اختصار یا کوتاه سازی شناسه ها اجتناب کنید، تا خوانایی برنامه افزایش یابد.



## مقدمه ای بر برنامه نویسی C++ \_\_\_\_\_ فصل دوم 11

### برنامه نویسی ایده‌ال



از ایجاد شناسه‌های که با یک خط زیرین یا دو خط زیرین آغاز می‌شوند اجتناب کنید، چراکه امکان دارد کامپایلر C++ از اسامی مشابه‌ای برای انجام مقاصد داخلی خود استفاده کرده باشد. در اینصورت از ایجاد تداخل مابین خود و کامپایلر جلوگیری خواهید کرد.

### اجتناب از خطا



زبان‌های همانند C++ همیشه در حال بسط و گسترش هستند. امکان در زمان تکامل، کلمات کلیدی جدیدی به آنها افزوده شود. از کلمات مسئولیت‌آوری همانند *object* بعنوان شناسه اجتناب کنید. حتی اگر امروز کلمه *object* جزء کلمات کلیدی در C++ نباشد، اما می‌تواند روزی تبدیل به کلمه کلیدی گردد و در آینده برنامه شما در زمان کامپایل با خطای جدیدی مواجه شود که قبلاً چنین چیزی وجود نداشت.

تقریباً می‌توان اعلان متغیرها را در هر کجای برنامه قرار داد، اما باید قبل از اینکه توسط برنامه بکار گرفته شوند، اعلان گردند. برای مثال، در برنامه شکل ۵-۲، اعلان در خط 9

```
int number1; // first integer to add
```

می‌توانست بلافاصله قبل از خط

```
std::cin >> number1; // read first integer from user into number1
```

قرار داده شود. اعلان موجود در خط 10

```
int number2; // second integer to add
```

می‌توانست بلافاصله قبل از خط 17

```
std::cin >> number2; // read second integer from user into number2
```

و اعلان خط 11

```
int sum; // sum of number1 and number2
```

می‌توانست بلافاصله قبل از خط 19

```
sum = number1 + number2; // add the numbers; store result in sum
```

اعلان گردد.

### برنامه نویسی ایده‌ال



بهبتر است در میان خطوط توضیحات از یک خط خالی استفاده شود. در اینصورت توضیحات از کد برنامه جدا شده و خوانایی برنامه افزایش می‌یابد.

### برنامه نویسی ایده‌ال



اگر ترجیح می‌دهید که اعلان‌ها را در ابتدای یک تابع قرار دهید، آنها را از بخش عبارات اجرایی موجود در



## 12 فصل دوم \_\_\_\_\_ مقدمه ای بر برنامه نویسی C++

داخل بدنه تابع و با استفاده از یک خط خالی جدا کنید تا این دو بخش از هم متمایز شوند.

خط 13

```
std::cout << "Enter first integer: "; // prompt user for data
```

مبادرت به چاپ رشته Enter first integer (بعنوان رشته لیترال نیز شناخته می شود) بر روی صفحه نمایش می کند. به چنین پیغام های *prompt* گفته می شود، چراکه به کاربر می گویند که چه کاری باید انجام دهد. خط 14

```
std::cin >> number1; // read first integer from user into number1
```

با استفاده از شی ورودی **cin** (از فضای نامی **std**) و عملگر **>>**، مقداری از صفحه کلید دریافت می کند. با استفاده از عملگر **>>** به همراه **std::cin** کاراکترها از یک ورودی استاندارد که معمولاً صفحه کلید است، دریافت می شوند. در واقع می توان گفت که **std::cin** مقداری برای **number1** دریافت می کند.

هنگامی که برنامه اجرا شده و به عبارت فوق می رسد، منتظر می ماند تا کاربر مقداری برای متغیر **number1** وارد سازد. کاربر با تایپ یک مقدار صحیح و سپس فشردن کلید *Enter* (گاهاً به این کلید، کلید *Return* هم می گویند) به برنامه پاسخ می دهد. با این عمل مقدار تایپ شده به کامپیوتر ارسال می شود. سپس کامپیوتر کاراکترهای دریافت شده را به یک عدد صحیح تبدیل و این عدد (یا مقدار) را به متغیر **number1** نسبت می دهد. از این نقطه به بعد هر مراجعه ای به **number1** در این برنامه مترادف با استفاده از این مقدار خواهد بود.

شی های **std::cout** و **std::cin** تعامل مابین کاربر و کامپیوتر را تسهیل می بخشند. بدلیل اینکه این تعامل بفرم یک گفتگو است، غالباً به اینحالت محاسبه تعاملی یا محاوره ای گفته می شود.

خط 16

```
std::cout << "Enter second integer: "; // prompt user for data
```

مبادرت به چاپ Enter second integer بر روی صفحه نمایش می کند. این عبارت به کاربر اعلان می کند که چه کاری انجام دهد. خط 17



## مقدمه‌ای بر برنامه‌نویسی C++ \_\_\_\_\_ فصل دوم 13

```
std::cin >> number2; // read second integer from user into number2
```

مقداری برای متغیر **number2** از سوی کاربر بدست می‌آورد.

عبارت تخصیص دهنده در خط 19

```
sum = number1 + number2; // add the numbers; store result in sum
```

مجموع متغیرهای **number1** و **number2** را محاسبه و نتیجه آنرا به متغیر **sum** تخصیص می‌دهد. از عملگر تخصیص = به این منظور استفاده شده است. مفهوم عبارت فوق به این مضمون است "sum مقدار خود را از **number1 + number2** بدست می‌آورد." اکثر محاسبات در بین عبارات تخصیصی صورت می‌گیرند. به عملگر = و عملگر +، عملگرهای باینری گفته می‌شود چرا که هر کدامیک از آنها دارای دو عملوند هستند. در مورد عبارت فوق، عملگر +، دارای دو عملوند **number1** و **number2** است. همچنین عملگر =، دارای دو عملوند **sum** و مقدار بدست آمده از **number1 + number2** است.

### برنامه‌نویسی ایده‌آل



در هر دو طرف یک عملگر باینری چند فاصله قرار دهید. فاصله‌ها باعث متمایز شدن نقش عملگر شده و خوانائی عبارت را افزایش می‌یابد.

خط 21

```
std::cout << "Sum is " << sum << std::endl; // display sum; end line
```

مبادرت به چاپ رشته کاراکتری **Sum is** و بدنبال آن مقدار عددی متغیر **sum** توسط عبارت **std::endl** می‌کند. **endl** کوتاه شده "end line" بوده و متعلق به فضای نامی **std** می‌باشد. کنترل کننده جریان **std::endl** یک خط جدید به خروجی منتقل و سپس "بافر خروجی را خالی می‌کند." به این معنی که، در برخی از سیستم‌ها خروجی در ماشین انباشته می‌شود تا زمانی که "ارزش" نمایش در صفحه نمایش را پیدا کنند، عبارت **std::endl** خروجی‌های انباشته شده را مجبور می‌کند تا در یکباره به نمایش در آیند.

دقت کند که عبارت قبلی دو مقدار از انواع مختلف را خارج می‌سازد. عملگر << از نحوه ارسال هر نوع داده به خروجی مطلع است. به روش استفاده از چندین عملگر << در یک عبارت، عملیات زنجیر کردن، متصل کردن یا فرآیند آبخاری گفته می‌شود. از اینرو، داشتن چندین عبارت خروجی برای خارج کردن چند داده متمایز ضروری نیست.



## 14 فصل دوم \_\_\_\_\_ مقدمه‌ای بر برنامه‌نویسی C++

البته می‌توان محاسبات را در بین عبارات خروجی هم انجام داد. می‌توانیم عبارات موجود در خطوط 19 و 21 را در یک عبارت، بفرم زیر داشته باشیم

```
std::cout << "Sum is " << number1 + number2 << std::endl;
```

بنابر این دیگر نیازی به متغیر `sum` نخواهد بود. براکت راست، `{`، به کامپیوتر اطلاع می‌دهد که به انتهای تابع `main` رسیده است.

از جمله توانمندیهای C++ این است که به کاربران اجازه می‌دهد تا نوع داده‌های مورد نیاز و متعلق به خود را ایجاد کنند (در فصل سوم در این مورد و در فصل‌های نهم و دهم نگاهی دقیق به این موضوع خواهیم داشت). سپس با نحوه رسیدگی C++ به هنگام کار با نوع داده‌های جدید با استفاده از عملگرهای `>>` و `<<` بیشتر آشنا خواهید شد (مبحث عملگرهای سربارگذاری در فصل یازدهم).

### ۲-۵ مفاهیم حافظه

اسامی متغیرها، همانند `number1`، `number2` و `sum` مطابق با مکان‌های واقعی در حافظه کامپیوتر هستند. هر متغیر دارای یک نام، نوع، سایز و مقدار است. در برنامه جمع ۲-۵ زمانیکه عبارت

```
std::cin >> number1; // read first integer from user into number1
```

در خط 14 اجرا می‌شود، کاراکترهای تایپ شده توسط کاربر به یک عدد صحیح تبدیل می‌شود و در مکانی از حافظه با نام `number1` و با کمک کامپایلر C++ قرار داده می‌شود. فرض کنید که کاربر عدد 45 را به عنوان مقداری برای `number1` وارد کند. کامپیوتر 45 را دریافت و در مکان `number1` همانند شکل ۲-۶ قرار می‌دهد.

زمانیکه مقداری در یک مکان حافظه قرار می‌گیرد، مقدار جدید بر روی مقدار قبلی بازنویسی می‌شود. مجدداً به سراغ برنامه جمع می‌رویم، هنگامی که عبارت

```
std::cin >> number2; // read second integer from user into number2
```

در خط 17 اجرا شود، فرض کنید کاربر مقدار 72 را وارد سازد، این مقدار وارد مکان `number2` شده و تصویر حافظه همانند شکل ۲-۷ خواهد شد. دقت کنید که این مکان‌های حافظه ضرورتاً، نبایستی در کنار هم قرار داشته باشند.

پس از اینکه برنامه مقادیر `number1` و `number2` را بدست آورد، این مقادیر را با هم جمع کرده و مجموع را در درون متغیر `sum` قرار می‌دهد. عبارت



## مقدمه ای بر برنامه نویسی C++ \_\_\_\_\_ فصل دوم 15

```
sum = number1 + number2; // add the numbers; store result in sum
```

علاوه بر آنکه عمل جمع را انجام می دهد، مقدار حاصله را هم در `sum` ذخیره می سازد. این عمل زمانی رخ می دهد که مجموع دو متغیر `number1` و `number2` محاسبه شده اند و در مکان `sum` ذخیره می شود. پس از اینکه `sum` محاسبه شد، حافظه بفرم شکل ۸-۲ تبدیل خواهد شد. توجه نمائید که مقادیر `number1` و `number2` پس از انجام محاسبه `sum` همچنان بدون تغییر باقی می ماند. با اینکه از این مقادیر استفاده شده است، اما مقادیر اولیه آنها با انجام عمل محاسباتی از بین نمی رود.

<code>number1</code>	45
----------------------	----

شکل ۶-۲ | مکان حافظه در حال نمایش نام و مقدار متغیر `number1`.

<code>number1</code>	45
<code>number2</code>	72

شکل ۷-۲ | مکان های حافظه پس از ذخیره سازی مقادیر برای `number1` و `number2`.

<code>number1</code>	45
<code>number2</code>	72
<code>sum</code>	117

شکل ۸-۲ | مکان های حافظه پس از محاسبه `sum` با استفاده از `number1` و `number2`.

### ۲-۶ محاسبات

اکثر برنامه ها محاسبات ریاضی انجام می دهند. عملگرهای ریاضی در جدول شکل ۹-۲ لیست شده اند. توجه کنید که تمام نمادهای بکار رفته در جبر در C++ بکار گرفته نمی شوند. علامت ستاره (\*) نشاندهنده ضرب و علامت درصد (%) نشاندهنده باقیمانده است. اکثر عملگرهای حسابی (در جدول ۹-۹



## 16 فصل دوم \_\_\_\_\_ مقدمه‌ای بر برنامه‌نویسی C++

۲) از نوع عملگرهای باینری هستند چرا که هر عملگر مابین دو عملوند قرار می‌گیرد. برای مثال، عبارت حسابی  $number1 + number2$  شامل عملگر باینری  $+$  و دو عملوند  $number1$  و  $number2$  می‌باشد.

عملیات C++	عملگر محاسباتی	عبارت جبری	عبارت C++
جمع	+	$f + 7$	$f + 7$
تفریق	-	$p - c$	$p - c$
ضرب	*	$bm$	$b * m$
تقسیم	/	$x / y$ یا $\frac{x}{y}$	$x / y$
باقیمانده	%	$x \div y$ $r \text{ mod } s$	$r \% s$

شکل ۹-۲ | عملگرهای محاسباتی.

### قوانین تقدم عملگر

C++ عملگرها را در عبارات محاسباتی با توالی که قانون تقدم عملگرها تعیین می‌کند بکار می‌برد. این قوانین شبیه قوانین موجود در جبر هستند:

۱- عملگرهایی که در درون جفت پرانتز قرار دارند دارای اولویت اول هستند. بنابر این برنامه‌نویس با استفاده از پرانتز می‌تواند ترتیب اجرای محاسبات را در دست بگیرد. پرانتزها دارای بالاترین سطح تقدم می‌باشند. در مواردی که پرانتزها به صورت تودرتو (آشپانه‌ای) قرار گرفته باشند، عملگر پرانتزی که در داخلی‌ترین سطح قرار دارد ابتدا انجام می‌گیرد، همانند

$$((a + b) + c)$$

عملگر قرار گرفته در جفت پرانتز داخلی ابتدا بکار گرفته می‌شود.

۲- عملگرهای ضرب و تقسیم و باقیمانده در مرحله بعدی بکار گرفته می‌شوند. اگر عبارتی شامل چندین عملگر ضرب، تقسیم و باقیمانده باشد، عملگرها از سمت چپ به راست اجرا خواهند شد. ضرب، تقسیم و باقیمانده دارای اولویت هم سطح یا برابر هستند.

۳- عملگرهای جمع و تفریق در آخرین مرحله به کار می‌روند. اگر عبارتی شامل چندین عملگر تفریق و جمع باشد، عملگرها به ترتیب از سمت چپ به راست اجرا خواهند شد. عملگرهای جمع و تفریق دارای اولویت هم سطح هستند.

وجود قوانین تقدم عملگرها، زبان C++ قادر می‌سازد تا عملگرها را با ترتیب صحیح بکار گیرد. در جدول شکل ۱۰-۲ خلاصه‌ای از قوانین تقدم عملگرها آورده شده است. این جدول با معرفی عملگرهای دیگر C++ در فصل‌های بعدی تکمیل تر خواهد شد. جدول کامل تقدم عملگرها در پیوست کتاب موجود است.





عملگر(ها)	عملیات	ترتیب ارزیابی (تقدم)
()	پرانتز	اولویت اول. اگر پرانتزها به صورت تودرتو باشند، عبارتی که در درون داخلی ترین پرانتز قرار دارد ابتدا محاسبه می شود. اگر چندین جفت پرانتز در یک خط قرار گرفته باشند (تودرتو نباشند) ترتیب اجرا از سمت چپ به راست خواهد بود.
*, /, %	ضرب و تقسیم و باقیمانده	اولویت دوم. اگر چند مورد از چنین عملگرهای وجود داشته باشد، ترتیب اجرا از سمت چپ به راست خواهد بود.
+, -	جمع و تفریق	اولویت آخر. اگر چندین عملگر جمع و تفریق وجود داشته باشد ترتیب اجرا از سمت چپ به سمت راست خواهد بود.

شکل ۱۰-۲ | تقدم عملگرهای محاسباتی.

### عبارات ساده جبری و C++

حال اجازه دهید تا به چند عبارت محاسباتی نگاهی بیاندازیم تا بخوبی با قوانین تقدم عملگرهای محاسباتی آشنا شوید. در هر مثالی که ذکر می شود عبارت جبری و معادل C++ آن عبارت نیز آورده شده است. مثال زیر یک عبارت ریاضی را نشان می دهد که منظور از آن به دست آوردن میانگین پنج عدد است:

$$\text{جبری: } m = \frac{a+b+c+d+e}{5}$$

$$\text{C++: } m = (a + b + c + d + e) / 5;$$

وجود پرانتز در این عبارت ضروری است چرا که عملگر تقسیم تقدم بالاتری نسبت به عملگر جمع دارد، در نتیجه مقدار داخلی پرانتز بر 5 تقسیم می شود. اگر پرانتز در این عبارت حذف شود، منظور محاسبه  $a + b + c + d + e / 5$  خواهد بود که معادل عبارت زیر است:

$$a + b + c + d + \frac{e}{5}$$

عبارت زیر نشاندهنده یک معادله است:

$$\text{جبری: } y = mx + b$$

$$\text{C++: } y = m * x + b;$$

وجود پرانتز در این عبارت نیاز نیست، چرا که عملگر ضرب تقدم بالاتری نسبت به عملگر جمع دارد و در ابتدا انجام می شود. عمل تخصیص در آخرین مرحله صورت می گیرد چرا که به نسبت عمل ضرب و جمع از اولویت پایین تری برخوردار است.

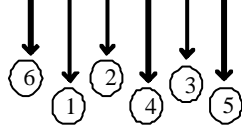
مثالی که در زیر آورده شده حاوی عملگرهای توان، ضرب، تقسیم اعشاری، جمع و تفریق است:



## 18 فصل دوم \_\_\_\_\_ مقدمه‌ای بر برنامه‌نویسی C++

جبری :  $z = pr\%q + w / x - y$

C++ :  $z = p * r \% q + w / x - y;$



دایره‌های حاوی اعداد نشان‌دهنده ترتیب اجرای عملگرها هستند. عملگر ضرب در اولویت اول قرار دارد و عملگرهای باقیمانده و تقسیم در اولویت‌های بعدی و به ترتیب از سمت چپ به راست اجرا می‌شوند و پس از آنها عملگرهای جمع و تفریق به ترتیب اجرا شده و در پایان عمل تخصیص صورت می‌گیرد.

توجه کنید که در این جدول در مورد پرانتزهای تودرتو مطالبی بیان شده است، اما تمام عبارات محاسباتی که دارای چندین جفت پرانتز هستند، ممکن است حاوی پرانتزهای تودرتو نباشد. برای مثال، اگر چه عبارت زیر

$$a * (b + c) + c * (d + e)$$

شامل دو جفت پرانتز است، اما هیچ کدامیک از آنها پرانتز تودرتو نمی‌باشند. در چنین حالتی هر دو آنها دارای سطح یکسان می‌باشند.

### ارزیابی معادله درجه دوم

برای درک بهتر قوانین تقدم عملگرها به مثال زیر که یک چند جمله‌ای درجه دوم است توجه کنید:

$$y = a * x * x + b * x + c;$$

دایره‌های حاوی اعداد نشان‌دهنده ترتیب اجرای عملگرها هستند. در C++ عملگر محاسباتی برای انجام عمل توان وجود ندارد، از اینرو عبارت  $x^2$  را بصورت  $x * x$  نشان داده‌ایم. بزودی در مورد تابع استاندارد کتابخانه‌ای  $pow$  (توان) صحبت خواهیم کرد.

### خطای برنامه‌نویسی



برخی از زبان‌های برنامه‌نویسی از عملگرهای  $**$  یا  $^$  برای نمایش توان استفاده می‌کنند. در حالیکه زبان C++ از

این عملگرها پشتیبانی نمی‌کند، و استفاده از آنها خطای نحوی خواهد بود.



## مقدمه ای بر برنامه نویسی C++ فصل دوم 19

حال فرض کنید که  $a, b, c$  و  $x$  بصورت  $a = 2, b = 3, c = 7$  و  $x = 5$  مقداردهی شده باشند. با توجه به شکل ۱۱-۲ تقدم عملگرها در این چند جمله‌ای درجه دوم و نتیجه اجرای آنرا تعقیب می‌کنیم.

### برنامه نویسی ایده‌آل



در عملیات جبری استفاده از پرانتزهای اضافی موجب می‌شود که عبارات از وضوح کافی برخوردار شوند (پرانتزهای غیرضروری *redundant parentheses* یا پرانتزهای افزونگی نیز نام‌یده می‌شوند). به کمک این پرانتزها می‌توان عبارات بزرگ و پیچیده را دسته‌بندی کرده و سبب واضح شدن طریقه انجام محاسبات شد.

### ۲-۷ تصمیم‌گیری: عملگرهای مقایسه‌ای و رابطه‌ای

در این قسمت به معرفی ساختار **if** در C++ می‌پردازیم که بر مبنای برقرار بودن یا نبودن برخی از شرط‌ها اقدام به تصمیم‌گیری می‌کند. عبارت موجود در یک ساختار **if** شرط نامیده می‌شود. اگر شرط مورد قبول واقع شود (شرط true باشد) عبارتی که در داخل بدنه ساختار **if** قرار گرفته‌اند اجرا می‌شوند و اگر شرط مورد قبول واقع نشود عبارات داخل بدنه اجرا نخواهند شد. برای آشنائی شما با چنین ساختاری به بررسی یک مثال خواهیم پرداخت.

شرط‌های که در ساختار **if** بکار می‌روند می‌توانند از عملگرهای مقایسه‌ای و رابطه‌ای که در جدول شکل ۱۲-۲ آورده شده‌اند، استفاده کنند. تمام عملگرهای رابطه‌ای دارای اولویت یکسان بوده و از سمت چپ به راست ارزیابی می‌شوند. عملگرهای مقایسه‌ای به نسبت عملگرهای رابطه‌ای از اولویت پایین‌تری برخوردار هستند و آنها هم از سمت چپ به راست ارزیابی می‌شوند.

گام اول  $Y = 2 * 5 * 5 + 3 * 5 + 7;$

$$2 * 5 = 10$$

سمت چپ‌ترین ضرب

گام دوم  $Y = 10 * 5 + 3 * 5 + 7;$

$$10 * 5 = 50$$

سمت چپ‌ترین ضرب

گام سوم  $Y = 50 + 3 * 5 + 7;$

$$3 * 5 = 15$$

ضرب قبل از جمع

گام چهارم  $Y = 50 + 15 + 7;$

$$50 + 15 = 65$$

سمت چپ‌ترین جمع

گام پنجم  $Y = 65 + 7;$

$$65 + 7 = 72$$

آخرین جمع

گام ششم  $Y = 72;$

شکل ۱۱-۲ | ترتیب اجرای عملگرهای محاسباتی در یک چند جمله‌ای درجه دوم.



مفهوم شرط	مثال در C++	عملگرهای مقایسه‌ای یا رابطه‌ای در C++	عملگرهای مقایسه‌ای یا رابطه‌ای استاندارد در جبر
			<b>عملگرهای مقایسه‌ای</b>
X با Y برابر است.	<code>x == y</code>	<code>==</code>	<code>=</code>
X با Y برابر نیست.	<code>x != y</code>	<code>!=</code>	<code>≠</code>
			<b>عملگرهای رابطه‌ای</b>
X از Y بزرگتر است.	<code>x &gt; y</code>	<code>&gt;</code>	<code>&gt;</code>
X از Y کوچکتر است.	<code>x &lt; y</code>	<code>&lt;</code>	<code>&lt;</code>
X بزرگتر یا مساوی Y است.	<code>x &gt;= y</code>	<code>&gt;=</code>	<code>≥</code>
X کوچکتر یا مساوی Y است.	<code>x &lt;= y</code>	<code>&lt;=</code>	<code>≤</code>

شکل ۱۲-۲ | عملگرهای مقایسه‌ای و رابطه‌ای.

### خطای برنامه‌نویسی



در صورتیکه مابین هر کدامیک از عملگرهای `==`، `!=` و `>` و `<` فاصله قرار دهید با خطای نحوی مواجه

خواهید شد.

### خطای برنامه‌نویسی



معکوس نوشتن عملگرهای `!=`، `>` و `<` بصورت `!=>`، `<=>` و `<=>` خطای نحوی بدانبال خواهد داشت. در برخی از

موارد نوشتن عملگر `!=` بصورت `! =` خطای نحوی تلقی نمی‌شود اما بصورت یک خطای منطقی و در زمان اجرای برنامه تاثیر

خود را نشان می‌دهد.

### خطای برنامه‌نویسی



اشتباه گرفتن رفتار عملگر رابطه‌ای `==` با عملگر تخصیص `=` می‌تواند خطای منطقی بدانبال داشته باشد.

برنامه زیر از شش عبارت `if` برای مقایسه بین دو عدد ورودی از سوی کاربر استفاده می‌کند. اگر شرط

موجود در هر کدامیک از عبارات `if` برقرار باشد (`true`)، خروجی مرتبط با آن عبارت به اجرا درخواهد

آمد. برنامه شکل ۱۳-۲ نشاندهنده برنامه و کادرهای ورودی و خروجی از اجرای نمونه برنامه است.

```

1 // Fig. 2.13: fig02_13.cpp
2 // Comparing integers using if statements, relational operators
3 // and equality operators.
4 #include <iostream> // allows program to perform input and output
5
6 using std::cout; // program uses cout
7 using std::cin; // program uses cin
8 using std::endl; // program uses endl
9
10 // function main begins program execution
11 int main()
12 {
13     int number1; // first integer to compare
14     int number2; // second integer to compare
15
16     cout << "Enter two integers to compare: "; // prompt user for data
17     cin >> number1 >> number2; // read two integers from user
18
19     if ( number1 == number2 )
20         cout << number1 << " == " << number2 << endl;
21

```



## مقدمه‌ای بر برنامه‌نویسی C++ \_\_\_\_\_ فصل دوم 21

```
22 if ( number1 != number2 )
23     cout << number1 << " != " << number2 << endl;
24
25 if ( number1 < number2 )
26     cout << number1 << " < " << number2 << endl;
27
28 if ( number1 > number2 )
29     cout << number1 << " > " << number2 << endl;
30
31 if ( number1 <= number2 )
32     cout << number1 << " <= " << number2 << endl;
33
34 if ( number1 >= number2 )
35     cout << number1 << " >= " << number2 << endl;
36
37 return 0; // indicate that program ended successfully
38
39 } // end function main
```

```
Enter two integers to compare: 3 7
3 != 7
3 < 7
3 <= 7
```

```
Enter two integers to compare: 22 12
22 != 12
22 > 12
22 >= 12
```

```
Enter two integers to compare: 7 7
7 == 7
7 <= 7
7 >= 7
```

شکل ۱۳-۲ | عملگرهای رابطه‌ای و مقایسه‌ای.

در خطوط 6-8

```
using std::cout; // program uses cout
using std::cin; // program uses cin
using std::endl; // program uses endl
```

از عبارات **using** استفاده شده است که نیاز به تکرار پیشوند **std::** را برطرف می‌کند. پس از بکارگیری عبارات **using**، می‌توانیم **cout** را بجای **std::cout**، **cin** را بجای **std::cin** و **endl** را بجای **std::endl** در مابقی برنامه بنویسیم. [توجه: از این نقطه به بعد در کتاب، در هر مثال از یک یا چند عبارت **using** استفاده شده است.]

برنامه‌نویسی ایده‌آل



ترجیحاً بلافاصله پس از **#include** از عبارت **using** استفاده کنید.



## مقدمه‌ای بر برنامه‌نویسی C++ فصل دوم

در خطوط 13-14

```
int number1; // first integer to compare
int number2; // second integer to compare
```

متغیرهای مورد نیاز برنامه اعلان شده‌اند. بخاطر دارید که متغیرها می‌توانند در یک خط یا چند خط اعلان شوند.

برنامه در خط 17 از روش آبرشاری برای دریافت داده به منظور دریافت دو عدد صحیح استفاده کرده است. بخاطر دارید که امکان نوشتن `cin` را به توجه به خط 7 فراهم آورده‌ایم (بجای `std::cin`). اولین مقدار قرائت شده و در متغیر `number1` قرار داده می‌شود و سپس مقدار دوم قرائت شده و در متغیر `number2` ذخیره می‌گردد.

ساختار `if` در خطوط 19-20

```
if ( number1 == number2 )
    cout << number1 << " == " << number2 << endl;
```

مبادرت به مقایسه متغیرهای `number1` و `number2` برای تست برابر بودن می‌کند. اگر مقادیر برابر باشند، عبارت موجود در خط 20 جمله مبنی بر اینکه اعداد با هم برابر هستند به نمایش در می‌آورد. اگر شرطی در یک یا چند ساختار `if` که در خطوط 23, 26, 29, 32 و 36 قرار دارند برقرار شود، عبارت متناظر توسط `cout` در خروجی به نمایش در می‌آید.

دقت کنید که هر ساختار `if` در برنامه شکل ۱۳-۲ دارای یک عبارت در بدنه خود بوده و بدنه تمام ساختارها بفرم دندانه‌دار نوشته شده‌اند. با دندانه‌دار نوشتن ساختار هر `if` وضوح و خوانایی برنامه را افزایش داده‌ایم. در فصل چهارم نشان خواهیم داد که چگونه می‌توان در ساختارهای `if` از چند عبارت استفاده کرد (به کمک جفت کاراکتر `{ }`).

### برنامه‌نویسی ایده‌آل



دندانه‌دار نوشتن عبارت یا عبارات موجود در درون ساختار `if` سبب می‌شود تا بدنه ساختار بخوبی آشکار شده و

بدنبال آن خوانایی برنامه افزایش یابد.

### برنامه‌نویسی ایده‌آل



نبایستی بیش از یک عبارت در هر خط برنامه قرار دهید.

### خطای برنامه‌نویسی



قرار دادن سیمکولن بلافاصله پس از شرط یک عبارت (پس از پرانتزها) ساختار `if` خطای منطقی بدنبال خواهد

داشت (اگرچه خطای نحوی به حساب نمی‌آید). سیمکولن سبب می‌شود، تا بدنه ساختار `if` خالی بنظر برسد، از اینرو ساختار `if` هیچ عملی انجام نمی‌دهد، صرفنظر از اینکه شرط برقرار باشد یا نباشد. علاوه بر این، بدنه اصلی ساختار `if` بصورت یک عبارت مجزا از `if` عمل می‌کند و همیشه اجرا شده و در اکثر مواقع نتایج اشتباه تولید می‌نماید.



## مقدمه ای بر برنامه‌نویسی C++ فصل دوم 23

به نحوه استفاده از فاصله‌ها در برنامه شکل ۱۳-۲ دقت کنید. در عبارات C++، کاراکترهای *white-space* همانند تب‌ها، خطوط جدید توسط کامپایلر در نظر گرفته نمی‌شوند. (اگر در درون رشته بکار گرفته شوند در نظر گرفته خواهند شد.) از اینرو، امکان دارد عبارات بر روی چند خط تقسیم شده و برطبق نظر برنامه‌نویس از هم فاصله پیدا کنند. جدا کردن هویت‌ها یا مشخصه‌ها، رشته‌ها (همانند "hello") و ثابت‌ها (همانند عدد 1000) بر روی چند خط خطای نحوی خواهد بود.

### خطای برنامه‌نویسی



جدا کردن یک مشخصه از هم بوسیله کاراکترهای *white-space* خطای نحوی بدنبال خواهد داشت (برای مثال

نوشتن *main* بصورت *ma in*).

### برنامه‌نویسی ایده‌آل



می‌توانید یک عبارت طولانی را بر روی چند خط قرار دهید. اگر می‌بایست یک عبارت به چند خط تقسیم شود، نقطه تقسیم را از مکانی همانند ویرگول در لیست ویرگول‌ها، یا پس از یک عملگر در عبارات طولانی قرار دهید.

در جدول شکل ۱۴-۲ تقدم عملگرهای معرفی شده در این فصل بطور یکجا آورده شده‌اند. اولویت عملگرها از بالا به پایین کاهش می‌یابد. دقت کنید که تمام این عملگرها بجز از عملگر تخصیص =، از سمت چپ به راست ارزیابی می‌شوند.

عملگر	شرکت‌پذیری	نوع
()	چپ به راست	پرانتز
*/%	چپ به راست	ضرب، تقسیم، باقیمانده
+ -	چپ به راست	جمع، تفریق
<< >>	چپ به راست	ورود و خروج داده
< <= > >=	چپ به راست	رابطه‌ای
== !=	چپ به راست	مقایسه‌ای
=	راست به چپ	تخصیصی

شکل ۱۴-۲ تقدم عملگرهای معرفی شده تا بدین مرحله.

### برنامه‌نویسی ایده‌آل



اگر عبارتی می‌نویسید که از عملگرهای متعددی تشکیل شده، بهتر است به جدول تقدم عملگرها مراجعه کنید. اگر از ترتیب عملگرها در عبارتی پیچیده مطمئن نیستید، از پرانتزها استفاده کرده و ترتیب اجرا را در دست بگیرید.

## ۲-۸ مبحث آموزشی مهندسی نرم‌افزار: بررسی نیازمندی‌های ATM

در این بخش طراحی و پیاده‌سازی شی‌گرا، مبحث آموزشی مهندسی نرم‌افزار را آغاز می‌کنیم. بخش‌های «مبحث آموزشی مهندسی نرم‌افزار» که در انتهای این فصل و چند فصل بعدی قرار داده شده‌اند، شما را به آسانی وارد بحث شی‌گرایی خواهند کرد. نرم‌افزاری برای یک سیستم ماشین تحویل‌دار خودکار ATM ایجاد خواهیم کرد، که تجربه مناسبی در زمینه طراحی و پیاده‌سازی برایتان به ارمغان خواهد آورد.



## 24 فصل دوم \_\_\_\_\_ مقدمه‌ای بر برنامه‌نویسی C++

در فصل‌های ۷-۳ و ۱۳، مراحل مختلفی از فرآیند طراحی شی‌گرا (OOD) را با استفاده از UML انجام خواهیم داد، و در کنار آن، مباحث مرتبط نیز در خود فصل‌ها مطرح می‌شوند. ضمیمه‌ای در ارتباط با پیاده‌سازی ATM با استفاده از تکنیک‌های برنامه‌نویسی شی‌گرا (OOP) در C++ آورده شده است. بحث ما یک بحث کاملاً آموزشی است، و حالت تمرینی ندارد و شما را کاملاً درگیر جزئیات کار با کد C++ می‌کند که پیاده‌سازی کننده برنامه هستند. این مطالب شما را با انواع مسائل قابل توجه در صنایع و همچنین راه‌حل‌های موجود آشنا خواهند کرد.

فرآیند طراحی را با معرفی مستند نیازمندی‌ها شروع می‌کنیم که تصریح‌کننده کل آنچیزی است که از یک سیستم ATM انتظار انجام آن را داریم و بطور دقیق آن را بررسی خواهیم کرد.

### مستند نیازها

فرض کنید یک بانک محلی مایل است تا یک سیستم ATM جدید را بکار گیرد و به کاربران (مشتریان بانک) اجازه دهد تا تعاملات مالی خود را با آن انجام دهند (شکل ۱۵-۲). هر کاربر می‌تواند فقط یک حساب در بانک داشته باشد. کاربران ATM باید قادر به دیدن موجودی حساب، برداشت از حساب و پس‌انداز باشند.

واسط کاربر ATM حاوی کامپونت‌های سخت‌افزاری زیر است:

- یک صفحه نمایش که پیغام‌ها را به کاربر بنمایش درمی‌آورد
- یک صفحه کلید که ورودی عددی را از کاربر دریافت می‌نماید
- تحویل‌دار خودکار که پول را به کاربر تحویل می‌دهد
- شکاف سپرده که پاکت سپرده را از کاربر تحویل می‌گیرد.

تحویل‌دار خودکار هر روز با پانصد عدد 20 دلاری پر می‌شود. [نکته: به این علت که این مبحث آموزشی است، برخی از عناصر مشخص ATM توصیف شده در اینجا، دقیقاً مطابق با ATM واقعی نیستند. برای مثال، معمولاً در یک ATM واقعی دستگاهی وجود دارد که شماره حساب مشتری را از یک کارت ATM می‌خواند، در حالیکه در ATM ما از کاربر خواسته می‌شود که شماره حساب خود را از طریق صفحه کلید وارد کند. همچنین در یک ATM واقعی قبض رسید در پایان هر عملیات یا جلسه چاپ می‌شود. امام تمام خروجی‌ها در این ATM بر روی صفحه نمایش ظاهر می‌شوند.]

شکل ۱۵-۲ | واسط کاربر ATM.





## مقدمه ای بر برنامه نویسی C++ \_\_\_\_\_ فصل دوم 25

بانک از شما می خواهد تا برنامه ای جدید برای انجام معاملات مالی مشتریان بانک از طریق ATM توسعه دهید. بانک بعداً نرم افزار را با سخت افزار ATM مجتمع خواهد کرد. نرم افزار بایستی عملکرد دستگاه های سخت افزاری (همانند پرداخت کننده پول، دریافت کننده سپرده) را در درون کامپونت های نرم افزاری کپسوله کند، اما نیازی ندارد که از عملکرد داخلی و جزئیات آنها مطلع باشد. فعلاً بخش سخت افزاری ATM تولید نشده است، از اینرو بجای نوشتن نرم افزاری که بروی ATM اجرا شود، باید نسخه اولیه از نرم افزار را برای اجرا بر روی یک کامپیوتر شخصی ایجاد کنید. این نسخه از برنامه، از ماینیور کامپیوتر برای شبیه سازی صفحه نمایش ATM و صفحه کلید کامپیوتر برای شبیه سازی، صفحه کلید ATM استفاده خواهد کرد.

یک جلسه ATM متشکل از تصدیق یا تایید کاربر (اثبات هویت کاربر) برپایه شماره حساب و شماره شناسایی شخصی (PIN) بوده و در ادامه معاملات مالی صورت می گیرد. برای تایید کاربر و انجام معاملات، بایستی ATM با پایگاه داده اطلاعات حساب بانکی در تعامل قرار گیرد. [نکته: پایگاه داده یک مجموعه سازماندهی شده از اطلاعات ذخیره شده در یک کامپیوتر است.] برای هر حساب بانکی، پایگاه داده یک شماره حساب، یک PIN و یک موجودی که نشان دهنده مقدار پول در آن حساب است، در خود ذخیره می سازد. [نکته: برای ساده تر شدن کار، فرض می کنیم که هدف بانک فقط داشتن یک دستگاه ATM است، بنابراین نیازی نیست که نگران نحوه دسترسی همزمان چندین ATM به این پایگاه داده باشید. علاوه بر این، فرض می کنیم که بانک هیچ تغییری در زمان استفاده کاربر (مشتری) از ATM، در پایگاه داده اعمال نمی کند. همچنین هر سیستم تجاری همانند ATM به دلایل قابل قبولی در ارتباط با مباحث امنیتی است که خارج از قلمرو تحصیلی یک دانشجوی ترم اول یا دوم کامپیوتر است.]

در اولین برخورد، مشتری با ATM باید توالی از رویدادهای زیر رخ دهند (به شکل ۱۵-۲ توجه نمائید):

۱- صفحه نمایش پیغام خوش آمدگویی (Welcome) را بنمایش درآورده و از کاربر می خواهد تا شماره حساب خود را وارد سازد.

۲- کاربر شماره حساب پنج رقمی خود را با استفاده از صفحه کلید وارد می سازد.

۳- در صفحه نمایش از کاربر خواسته می شود تا PIN را وارد سازد، که مرتبط با شماره حساب است.

۴- کاربر از طریق صفحه کلید، PIN پنج رقمی خود را وارد می سازد.



## 26 فصل دوم \_\_\_\_\_ مقدمه ای بر برنامه نویسی C++

۵- اگر شماره حساب و PIN ورودی معتبر باشند، ظاهر صفحه نمایش همانند شکل ۱۶-۲ بوده و منوی اصلی در آن ظاهر می‌گردد. اگر شماره حساب یا PIN اشتباه باشد، پیغام مناسب در صفحه نمایش ظاهر شده و ATM به مرحله اول باز می‌گردد تا فرآیند تایید را از نو آغاز کند.

پس از تایید کاربرد از سوی ATM، منوی اصلی (شکل ۱۶-۲) لیستی از گزینه‌های عددی برای هر سه نوع تراکنش بنمایش در می‌آورد: درخواست موجودی (گزینه ۱)، برداشت (گزینه ۲) و سپرده‌گذاری (گزینه ۳). همچنین منوی اصلی، گزینه‌ای برای خروج از سیستم در اختیار کاربر قرار می‌دهد (گزینه ۴). پس از نمایش منوی اصلی، کاربر می‌تواند تراکنش موردنظر خود را از طریق وارد کردن 1، 2، 3 یا خروج از سیستم، 4 انتخاب کند. اگر کاربر گزینه اشتباهی را وارد سازد، پیغامی به نمایش درخواهد آمد، و سپس مجدداً منوی اصلی بنمایش در می‌آید.

اگر کاربر گزینه 1 را انتخاب کند (با وارد کردن عدد 1) تا از میزان موجودی خود مطلع گردد، این امر صورت خواهد گرفت. برای انجام این کار، بایستی ATM میزان موجودی را از پایگاه داده بانک بازیابی کند.

مراحل زیر زمانی رخ می‌دهند که کاربر گزینه 2 را برای برداشت پول انتخاب کرده باشد:

۶- منوی در صفحه نمایش ظاهر می‌شود (شکل ۱۷-۲) که حاوی مقادیر استاندارد قابل پرداخت است. (گزینه 1) \$20، (گزینه 2) \$40، (گزینه 3) \$60، (گزینه 4) \$100 و (گزینه 5) \$200. همچنین این منو دارای یک گزینه برای لغو تراکنش کاربر است (گزینه 6).

### شکل ۱۷-۲ | منوی برداشت پول ATM.

۷- کاربر با استفاده از صفحه کلید، انتخاب خود را انجام می‌دهد (گزینه‌های ۱-۶).

۸- اگر مقدار درخواستی برای برداشت، بیشتر از میزان موجودی کاربر باشد، پیغامی این مطلب را به عرض کاربر رسانده و از وی می‌خواهد که مقدار کمتری تقاضا کند. سپس ATM به مرحله اول باز می‌گردد. اگر مقدار درخواستی کمتر از موجودی یا برابر آن باشد (یک مقدار قابل قبول)، ATM مرحله 4 را آغاز خواهد کرد. اگر کاربر مبادرت به لغو تراکنش کند (گزینه 6)، ATM منوی اصلی را به نمایش در آورده و منتظر ورودی کاربر می‌شود (شکل ۱۶-۲).

۹- اگر پرداخت‌کننده خودکار پول، به میزان کافی پول برای برآورده کردن تقاضای مشتری داشته باشد، ATM وارد مرحله ۵ خواهد شد. در غیر اینصورت، صفحه نمایش پیغامی مبنی بر اینکه میزان پول



## مقدمه ای بر برنامه نویسی C++ \_\_\_\_\_ فصل دوم 27

دستگاه کمتر از مقدار درخواستی است از کاربر می خواهد که مقدار کمتری را انتخاب نماید. سپس ATM به مرحله اول بازمی گردد.

۱۰- ATM مقدار پول برداشتی را از موجودی حساب کاربر در پایگاه داده بانک کم می کند.

۱۱- پرداخت کننده خودکار، مقدار پول درخواستی را به کاربر تحویل می دهد.

۱۲- پیغامی در صفحه نمایش ظاهر شده و به کاربر یادآوری می کند، پول را بردارد.

مراحل زیر زمانی رخ می دهند که کاربر عدد 3 را از منوی اصلی به منظور سپرده گذاری انتخاب کرده باشد:

۱۳- در صفحه نمایش، به کاربر اعلان می شود که مقدار سپرده گذاری خود را وارد سازد یا برای لغو تراکنش صفر را وارد کند.

۱۴- کاربر از طریق صفحه کلید، مقدار سپرده گذاری یا صفر را وارد می سازد [نکته: صفحه کلید دارای نقطه اعشار یا نماد دلار نمی باشد، از اینرو، کاربر نمی تواند یک مقدار دلاری واقعی همانند \$1.25 را وارد سازد. بجای آن، کاربر باید مقدار سپرده خود را بعنوان یک عدد از سنت ها وارد کند (مثلاً 125). سپس ATM این عدد را بر 100 تقسیم می کند، تا عددی که نشان دهنده مقدار دلاری است بدست آید (مثلاً  $1.25 = 125 \div 100$ ).

۱۵- اگر کاربر میزان سپرده را مشخص سازد، ATM به مرحله ۴ می رود. اگر کاربر، مبادرت به لغو تراکنش کند (با وارد کردن صفر)، ATM منوی اصلی را به نمایش درآورده و در انتظار ورودی کاربر باقی می ماند (شکل ۱۶-۲)

۱۶- در صفحه نمایش، پیغامی به کاربر اعلان می کند که پاکت سپرده را در شکاف سپرده قرار دهد.

۱۷- اگر شکاف سپرده، پاکت سپرده را در عرض دو دقیقه دریافت نماید، ATM مبادرت به افزایش اعتبار کاربر، در میزان موجودی وی در پایگاه داده بانک می کند [نکته: این مقدار پول، بلافاصله برداشت نمی شود. ابتدا باید بانک به لحاظ فیزیکی مبادرت به بازبینی پول نقد و چک های موجود در پاکت کرده و پس از تایید بانک، حساب کاربر در پایگاه داده به روز می شود. این عملیات مستقل از سیستم ATM صورت می گیرد.] اگر در مدت زمان مشخص شده، شکاف سپرده، پاکتی دریافت نکند، پیغامی در صفحه نمایش مبنی بر لغو تراکنش از طرف سیستم ظاهر خواهد شد. سپس ATM منوی اصلی را به نمایش درآورده و منتظر ورودی کاربر باقی ماند.



پس از اجرای موفقیت‌آمیز یک تراکنش توسط سیستم، باید مجدداً منوی اصلی بنمایش درآید (شکل ۱۶-۲) تا کاربر قادر به انجام تراکنش‌های دیگر باشد. اگر کاربر گزینه ۴ را انتخاب کند (خروج از سیستم)، پیغام تشکر بنمایش درآمده و سپس پیغام خوش‌آمدگویی، برای کاربر بعدی بنمایش درمی‌آید.

### تحلیل سیستم ATM

بخش قبلی مثال ساده شده‌ای از یک مستند نیازمندی‌ها بود. بطور نمونه، چنین مستندی، نتیجه‌ای از نیازمندی‌های جمع‌آوری شده است که شامل مصاحبه با کاربران اصلی سیستم و متخصصین در فیلدهای مربوط به سیستم است. برای مثال، تحلیل‌گر سیستم کسی است که برای آماده کردن مستند نیازمندی‌های نرم‌افزار بانک استخدام می‌شود (مثلاً سیستم ATM که در اینجا توضیح داده شده است) و می‌تواند با متخصصین امور مالی مصاحبه انجام دهد تا درک دقیقی از کاری که نرم‌افزار باید انجام دهد بدست آورد. تحلیل‌گر از اطلاعات بدست آمده، لیستی از نیازهای سیستم جمع‌آوری می‌کند تا طراحان سیستم را بخوبی راهنمایی کند.

فرآیند جمع‌آوری اطلاعات نیازمندی‌ها، یک وظیفه کلیدی در مرحله اول چرخه زندگی نرم‌افزار است. چرخه زندگی نرم‌افزار، تصریح‌کننده مراحل است که نرم‌افزار از بدو تولد تا زمان بازنشستگی طی می‌کند. بطور نمونه این مراحل عبارتند از: تحلیل، طراحی، پیاده‌سازی، تست و خطایابی، استفاده، نگهداری و بازنشستگی. چندین مدل برای چرخه طول عمر نرم‌افزار وجود دارد که هر یک دارای مزایا و مشخصات خاص بوده که روش انجام مراحل مختلف را به مهندسان نرم‌افزار گوشزد می‌کنند. مدل آبشاری (waterfall model) هر مرحله را یکی بعد از دیگری انجام می‌دهد، در حالیکه مدل تکرارکننده (iterative model) می‌تواند مراحل را یک یا چندین بار در فرآیند چرخه طول عمر یک محصول تکرار نماید.

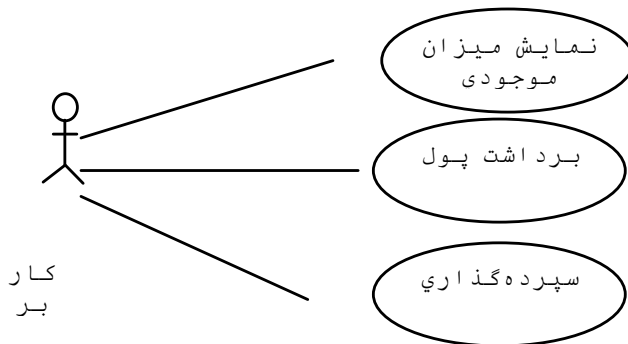
مرحله تحلیل در چرخه طول عمر نرم‌افزار، متمرکز بر تعریف مسئله برای حل کردن آن است. به هنگام طراحی هر سیستمی، باید مسئله بدرستی حل شود (راه حل باید صحیح باشد). تحلیل‌گران سیستم مبادرت به جمع‌آوری نیازهایی می‌کنند که قادر به حل مسئله مشخصی هستند. مستند نیازها که آن را در اینجا برای سیستم ATM مطرح کرده‌ایم، بقدر کافی گویا است و نیازی نیست که وارد یک مرحله تحلیل اضافی تر شوید.

برای ثبت اینکه سیستم مورد نظر چه کاری باید انجام دهد، غالباً توسعه‌دهندگان از تکنیکی بنام مدل‌سازی use case (حالت استفاده) کمک می‌گیرند. این فرآیند حالات مورد استفاده از سیستم را معین می‌سازد، که هر یک نشانگر یک قابلیت مختلف است که سیستم در اختیار سرویس‌گیرندگان خود قرار



## مقدمه ای بر برنامه نویسی C++ فصل دوم 29

می‌دهد. برای مثال، بطور نمونه ATMها دارای چندین حالت استفاده همانند «نمایش میزان موجودی»، «برداشت پول»، «سپرده گذاری»، «انتقال پول مابین حساب‌ها» و «خرید تمبر پستی» هستند. سیستم ATM ساده شده که قصد ساخت آن را داریم، فقط دارای سه حالت استفاده است که در شکل ۱۸-۲ دیده می‌شود.



شکل ۱۸-۲ | دیاگرام حالت استفاده برای سیستم ATM از منظر کاربر.

هر حالت استفاده توصیف کننده یک سناریو است که کاربر از سیستم استفاده می‌کند. در حال حاضر حالات استفاده از سیستم ATM را از مستند نیازها مطالعه کرده‌اید، که هر مرحله نیازمند نوعی از تراکنش است (نمایش موجودی، برداشت پول و سپرده گذاری)، که آنها را در سه حالت استفاده از ATM قرار داده‌ایم.

### دیاگرام‌های Use Case

در این بخش مبادرت به معرفی یکی از چندین دیاگرام UML برای ATM مطرح شده در این مبحث آموزشی می‌کنیم. یک دیاگرام حالت استفاده (use case) برای مدل کردن تراکنش‌های مابین سرویس گیرندگان سیستم (در این مورد، مشتریان بانک مدنظر هستند) و سیستم ایجاد می‌کنیم. هدف، نمایش انواع تراکنش‌های کاربران با سیستم است بدون اینکه جزئیات را وارد کار نمائیم (جزئیات در دیاگرام‌های دیگر UML تدارک دیده می‌شوند، که در ادامه با آنها مواجه خواهید شد). غالباً دیاگرام‌های حالت استفاده همراه با جملات غیررسمی که توصیف کننده دقیق تر حالات استفاده هستند، همراه می‌باشند، همانند جملاتی که در مستند نیازها دیده می‌شود. دیاگرام‌های حالت استفاده در مرحله تحلیل چرخه عمر نرم افزار تولید می‌شوند. در سیستم‌های بزرگتر، دیاگرام‌های حالت استفاده ساده هستند اما جزء



## 30 فصل دوم \_\_\_\_\_ مقدمه‌ای بر برنامه‌نویسی C++

ابزارهای ضروری هستند که به طراحان سیستم کمک می‌کنند تا تمرکز خود را بر روی نیازهای کاربران حفظ کنند.

شکل ۱۸-۲ نمایشی از دیاگرام حالت استفاده برای سیستم ATM است. تصویر آدمک نشان‌دهنده یک بازیگر (*actor*) است که تعریف‌کننده نقش‌های است که یک موجودیت خارجی همانند یک شخص یا سیستم دیگر، به هنگام در تعامل قرار گرفتن با سیستم ایفاء می‌کند. برای سیستم ATM ما، بازیگر کاربری است که می‌تواند میزان موجودی را مشاهده کند، از سیستم پول دریافت و در آن سپرده‌گذاری انجام دهد. کاربر یک شخص حقیقی نیست، اما دارای نقش‌های از یک شخص واقعی است (زمانیکه نقشی از یک کاربر را باز می‌کند) و می‌تواند آن نقش‌ها را در زمان تعامل با ATM بازی کند. دقت کنید که دیاگرام حالت استفاده می‌تواند حاوی چندین بازیگر باشد. برای مثال، دیاگرام حالت استفاده در یک سیستم ATM واقعی، می‌تواند شامل یک بازیگر بنام Administrator (مدیر) باشد که مسئول پر کردن هر روز پول در تحویل‌دار است.

شناسایی بازیگر در سیستم را با بررسی مستند نیازها مشخص می‌کنیم که عبارت است از "کاربران ATM بایستی قادر به مشاهده میزان موجودی، برداشت پول و سپرده‌گذاری باشند." از اینرو، بازیگر در هر یک از این سه حالت استفاده، کاربری است که با ATM در تعامل قرار می‌گیرد. موجودیت خارجی، یک شخص حقیقی، بخشی از نقش کاربر را در انجام تراکنش‌های مالی بازی می‌کند. در شکل ۱۸-۲ یک بازیگر بنام «کاربر» نشان داده شده است. UML هر کدامیک از حالات استفاده را بصورت یک بیضی متصل به بازیگر توسط یک خط ساده را مدل می‌کند.

بایستی مهندسان نرم‌افزار یا بطور دقیق‌تر طراحان سیستم، مبادرت به تحلیل مستند نیازها یا تنظیم حالات استفاده و طراحی سیستم قبل از شروع به برنامه‌نویسی با یک زبان برنامه‌نویسی خاص کنند. در مدت زمان مرحله تحلیل، طراحان سیستم بر درک مستند نیازها تمرکز دارند تا مشخصاتی با معیار بالا بدست آید که توصیف‌کننده آنچه که سیستم باید انجام دهد، باشد. خروجی مرحله طراحی (طراحی مشخصه‌ها) بایستی بقدر کافی واضح و شفاف باشد که نحوه ایجاد سیستم را برای برآورده کردن نیازها بیان کند. در چند بخش بعدی «مبحث آموزشی مهندسی نرم‌افزار» این مراحل را از طریق طراحی شی‌گرا (OOD) بر روی سیستم ATM به منظور تولید طرحی از مشخصه‌ها انجام خواهیم داد که حاوی مجموعه‌ای از دیاگرام‌های UML و جملات پشتیبانی‌کننده است. بخاطر دارید که UML برای استفاده در هر فرآیند OOD طراحی شده است. چندین پردازش‌کننده وجود دارد که یکی از بهترین آنها RUP (Rational Unified Process) است که توسط شرکت Rational Software توسعه پیدا کرده است (این



## مقدمه ای بر برنامه نویسی C++ فصل دوم 31

شرکت تبدیل به بخشی از IBM شده است). RUP یک پردازش کننده بسیار مناسب و مطلوب برای طراحی برنامه های کاربردی در سطح صنایع است. برای این مبحث آموزشی، فرآیند طراحی ساده شده خود را معرفی می کنیم.

### طراحی سیستم ATM

در این بخش وارد مرحله طراحی سیستم ATM می شویم. سیستم مجموعه ای از کامپونت ها است که برای حل مسئله ای با هم در تعامل قرار می گیرند. برای مثال، برای اینکه سیستم ATM وظایف تعیین شده را انجام دهد، باید دارای یک واسط کاربر بوده (شکل ۱۵-۲) و حاوی نرم افزاری باشد که قادر به انجام تراکنش های مالی و کار با پایگاه داده اطلاعات حساب مشتریان در بانک باشد. ساختار سیستم، توصیف کننده شی های سیستم و روابط داخلی آنها است. رفتار سیستم توصیف کننده نحوه تغییر عملکرد شی های سیستم و برقراری روابط بین آنها است. هر سیستمی دارای ساختار و رفتار است، که طراحان باید آنها را مشخص سازند. چندین نوع مشخص از ساختار و رفتار سیستم وجود دارد. برای مثال، تعامل مابین شی ها در یک سیستم، متفاوت از تعامل مابین سیستم و کاربر است، با این همه هنوز هر دو بخشی از رفتار سیستم محسوب می شوند. نسخه 2 UML تصریح کننده 13 نوع دیاگرام برای مستند کردن مدل های سیستم است. هر دیاگرام صفات مشخصی از ساختار سیستم یا رفتار آن را مدل سازی می کند، شش دیاگرام در ارتباط با ساختار سیستم بوده و هفت دیاگرام باقیمانده مربوط به رفتار سیستم هستند. در اینجا فقط شش نوع دیاگرام بکار رفته در این مبحث را لیست کرده ایم، یکی از آنها بنام دیاگرام کلاس، مبادرت به مدل کردن ساختار سیستم می کند و مابقی در ارتباط با مدل سازی رفتار سیستم هستند.

۱- دیاگرام حالت استفاده، همانند شکل ۱۸-۲، مبادرت به مدل سازی تعامل صورت گرفته مابین سیستم و موجودیت خارجی آن (بازیگران) با جملات حالت استفاده می کند (قابلیت های سیستم همانند «نمایش میزان موجودی»، «برداشت پول» و «سپرده گذاری»).

۲- دیاگرام های کلاس، که در بخش ۱۱-۳ با آنها آشنا خواهید شد. این دیاگرام ها در مدل کردن کلاس ها یا «ایجاد بلوک های» مورد استفاده در سیستم کاربرد دارند. هر اسم یا «چیز» توصیف شده در مستند نیازها، نامزد تبدیل شدن به یک کلاس در سیستم است (همانند «حساب»، «صفحه کلید»). دیاگرام های کلاس در مشخص کردن روابط ساختاری موجود مابین اجزای سیستم نقش دارند. برای مثال، دیاگرام کلاس سیستم ATM مشخص می کند که ATM به لحاظ فیزیکی متشکل از یک صفحه نمایش، صفحه کلید، تحویل دار خود کار پول و شکاف سپرده گذاری است.



## 32 فصل دوم \_\_\_\_\_ مقدمه‌ای بر برنامه‌نویسی C++

۳- دیاگرام‌های وضعیت ماشین، که در بخش ۱۱-۳ با آنها آشنا خواهید شد. این دیاگرام‌ها در مدل‌سازی روش‌های که یک شی تغییر وضعیت یا حالت می‌دهد کاربرد دارند. وضعیت یک شی توسط مقادیری که از صفات شی در زمان اجرا بدست می‌آیند، تعیین می‌شود. زمانیکه وضعیت یک شی تغییر پیدا می‌کند، امکان دارد شی رفتار متفاوتی در سیستم بخود بگیرد. برای مثال، پس از اعتبارسنجی PIN کاربر، تراکنش ATM از وضعیت «کاربر تایید نشده» به وضعیت «کاربر تایید شده» تبدیل شده و در این لحظه ATM به کاربر اجازه می‌دهد تا تراکنش‌های مالی انجام دهد (مشاهده میزان موجودی، برداشت پول و سپرده‌گذاری).

۴- دیاگرام‌های فعالیت، که در بخش ۱۱-۵ با آنها آشنا خواهید شد. این دیاگرام‌ها در مدل‌سازی فعالیت یک شی کاربرد دارند (جریان کار یا روند کار یک شی (توالی از رویدادها) در مدت زمان اجرای برنامه). یک دیاگرام فعالیت مبادرت به مدل کردن اعمال یک شی و همچنین ترتیب انجام این اعمال را مشخص می‌نماید. برای مثال، یک دیاگرام فعالیت نشان می‌دهد که ATM باید میزان موجودی حساب کاربر را (از پایگاه داده اطلاعات حساب) قبل از اینکه صفحه نمایش موجودی را بنمایش درآورد، تهیه نماید.

۵- دیاگرام‌های ارتباطی (در نسخه‌های قبلی UML این دیاگرام، دیاگرام‌های همکاری نامیده می‌شود) مدل‌کننده تعامل‌های صورت گرفته مابین شی‌های یک سیستم هستند، با تاکید بر اینکه کدام تعامل رخ دهد. در بخش ۱۲-۷ با این نوع دیاگرام‌ها آشنا خواهید شد. برای مثال باید ATM با پایگاه داده حساب بانکی ارتباط برقرار کند تا میزان موجودی را بازیابی نماید.

۶- دیاگرام‌های توالی، این دیاگرام‌ها نیز مبادرت به مدل‌سازی تعامل‌های صورت گرفته مابین شی‌های یک سیستم می‌کنند، اما برخلاف دیاگرام‌های ارتباطی، تاکید آنها بر زمان رخ دادن تعامل‌ها است. در بخش ۱۲-۷ با این نوع دیاگرام‌ها آشنا خواهید شد. برای مثال، صفحه نمایش به کاربر اطلاع می‌دهد که مقدار پولی که می‌خواهد برداشت کند را قبل از پرداخت وارد نماید.

در بخش ۱۱-۳ به ادامه طراحی ATM با شناسایی کلاس‌ها از طریق مستند نیازها ادامه خواهیم داد. این کار را با استخراج اسامی کلیدی و تعبیر اسامی از مستند نیازها انجام خواهیم داد. با استفاده از این کلاس‌ها، اولین پیش‌نویس خود را از دیاگرام کلاس ایجاد می‌کنیم که ساختار سیستم ATM را مدل‌سازی می‌کند.

### اینترنت و منابع وب

[www.306.ibm.com/software/rational/uml/](http://www.306.ibm.com/software/rational/uml/)





[www.softdocwiz.com/Dictionary.htm](http://www.softdocwiz.com/Dictionary.htm)  
[www-306.ibm.com/software/rational/offerings/design.html](http://www-306.ibm.com/software/rational/offerings/design.html)  
[www.embarcadero.com/products/describe/index.html](http://www.embarcadero.com/products/describe/index.html)  
[www.borland.com/together/index.html](http://www.borland.com/together/index.html)  
[www.ilogix.com/rhapsody/rhapsody.cfm](http://www.ilogix.com/rhapsody/rhapsody.cfm)  
[argouml.tigris.org](http://argouml.tigris.org)  
[www.objectsbydesign.com/booklist.html](http://www.objectsbydesign.com/booklist.html)  
[www.objectsbydesign.com/tools/umltools-bycompany.html](http://www.objectsbydesign.com/tools/umltools-bycompany.html)  
[www.ootips.org/ood-principles.html](http://www.ootips.org/ood-principles.html)  
[www.cetus-link.org100-uml.html](http://www.cetus-link.org100-uml.html)  
[www.agilemodeling.com/essays/umlDiagrams.htm](http://www.agilemodeling.com/essays/umlDiagrams.htm)

### کتابهای توصیه شده

کتابهای معرفی شده در این بخش حاوی اطلاعاتی در ارتباط با طراحی شی گرا با UML هستند.

- Booch, G. Object-Oriented Analysis and Design with Applications, Third Edition. Addison-Wesley, 2004.
- Eriksson, H., et al. UML2 Toolkit. New York: John Wiley, 2003.
- Kruchten, P. The Rational Unified Process: An Introduction. Boston: Addison-Wesley, 2004.
- Lormann, C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design, Second Edition. Upper Saddle River, NJ: Prentice Hall, 2002.
- Roques, P. UML in Practical: The Art of Modeling Software Systems Demonstrated through worked Examples and solutions. New York: John Wiley, 2004.
- Rosenberg, D., and K. Scott. Applying use Case Driven Object Modeling with UML: An Annotated e-Commerce Example. Reading, MA: Addison-Wesley, 2001.
- Rumbaugh, J., I. Jacobson and G. Booch. The Complete UML Training Course. Upper Saddle River, NJ: Prentice Hall, 2000.
- Rumbaugh, J., I. Jacobson and G. Booch. The unified Modeling Language Reference Manual. Reading, MA: Addison-Wesley, 1999.
- Rumbaugh, J., I. Jacobson and G. Booch. The unified Software Development Process. Reading, MA: Addison-Wesley, 1999.

### خودآزمایی مبحث آموزشی مهندسی نرم افزار

۲-۱ فرض کنید که سیستم ATM مورد نظر ما قادر به انتقال پول بین دو حساب بانکی است. در دیاگرام حالت استفاده شکل ۲-۱۸ این تغییر را اعمال کنید.

۲-۲ \_\_\_\_\_، مدل کننده تعامل مابین شی ها در یک سیستم با تاکید بر زمان رخ دادن تعامل ها است.

(a) دیاگرام های کلاس

(b) دیاگرام های توالی

(c) دیاگرام های ارتباطی

(d) دیاگرام های فعالیت

۲-۳ کدامیک از لیست های زیر نشان دهنده چرخه عمر صحیح نرم افزار هستند؟

(a) طراحی، تحلیل، پیاده سازی، تست

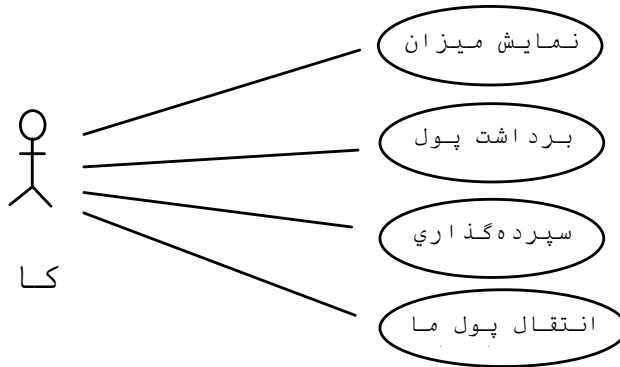


## 34 فصل دوم \_\_\_\_\_ مقدمه ای بر برنامه نویسی C++

- (b) طراحی، تحلیل، تست، پیاده سازی
- (c) تحلیل، طراحی، تست، پیاده سازی
- (d) تحلیل، طراحی، پیاده سازی، تست

### پاسخ خودآزمایی مبحث آموزشی مهندسی نرم افزار

شکل ۲-۱۹ حاوی دیاگرام حالت استفاده اصلاح شده از سیستم ATM است که به کاربر امکان انتقال پول مابین حسابها را فراهم می آورد.



شکل ۲-۱۹ | دیاگرام حالت برای نسخه ای از ATM که قادر به انجام انتقال پول مابین حسابها هم می باشد.

b ۲-۲

d ۲-۳

### خودآزمایی

۲-۱ جاهای خالی را در عبارات زیر با کلمات مناسب پر کنید:

- (a) هر برنامه C++ اجرای خود را از تابع ..... آغاز می کند.
  - (b) بدنه هر تابع با کاراکتر ..... شروع و با کاراکتر ..... به پایان می رسد.
  - (c) هر عبارتی با کاراکتر ..... به پایان می رسد.
  - (d) کاراکتر توالی `\n` نشاندهنده کاراکتر ..... است، که باعث می شود تا کرسر به ابتدای خط بعدی در صفحه نمایش منتقل شود.
  - (e) از عبارت ..... برای تصمیم گیری استفاده می شود.
- ۲-۲ کدامیک از عبارات زیر صحیح و کدامیک اشتباه است. اگر عبارتی اشتباه است علت آنرا توضیح دهید. فرض کنید از عبارت `using std::cout;` استفاده شده است.



## مقدمه ای بر برنامه نویسی C++ \_\_\_\_\_ فصل دوم 35

(a) توضیحات سبب می شوند تا کامپیوتر مبادرت به چاپ عبارت قرار گرفته پس از // بر روی صفحه نمایش به هنگام اجرای برنامه کند.

(b) کاراکتر توالی n به هنگام کار با cout موجب می شود تا کرسر به ابتدای خط بعدی در صفحه نمایش منتقل شود.

(c) قبل از اینکه بتوان از متغیری استفاده کرد باید آن را اعلان کرده باشیم.

(d) به هنگام اعلان متغیرها بایستی نوع آنها تعریف شود.

(e) در نظر C++ متغیرهای number و NuMbEr یکسان هستند.

(f) تقریباً می توان اعلان ها را در هر کجای بدنه یک تابع C++ قرار داد.

(g) از عملگر % فقط می توان در کنار عملوندهای صحیح استفاده کرد.

(h) تمام عملگرهای محاسباتی +, %, /, \* و - دارای اولویت برابر هستند.

(i) یک برنامه C++ که سه خط در خروجی چاپ می کند بایستی حاوی سه عبارت خروجی با بکارگیری cout و عملگر درج باشد.

۳-۲ یک عبارت تک جمله ای در C++ بنویسید که موارد مورد تقاضا را برآورده سازد: (فرض کنید که از عبارت using استفاده نشده است)

(a) متغیرهای c, thisIsAvariable, q76354 و number را از نوع int اعلان کنید.

(b) به کاربر اعلان کنید تا یک عدد صحیح وارد کند. در انتهای پیغام یک کاراکتر کولن (: ) و سپس یک فاصله قرار دهید.

(c) مقدار صحیح وارد شده از طریق صفحه کلید را دریافت و آنرا در متغیر age ذخیره سازد.

(d) اگر متغیر number برابر 7 نباشد، عبارت "The variable number is not equal to 7" چاپ شود.

(e) پیغام "This is a C++ program" در یک خط چاپ شود.

(f) پیغام "This is a C++ program" را دو خط چاپ کند. بطوریکه خط اول با C++ به پایان برسد.

(g) پیغام "This is a C++ program" را به نحوی چاپ کند، که هر کلمه در خط مجزائی چاپ شود.

(h) پیغام "This is a C++ program" را به نحوی چاپ کند، که هر کلمه با کلمه دیگر به اندازه یک tab فاصله داشته باشد.

۴-۲ یک عبارت (یا توضیح) برای برآورده کردن موارد زیر بنویسید (فرض کنید که از عبارت using استفاده شده است):

(a) نشان دهد که برنامه مبادرت به ضرب سه عدد صحیح می کند.

(b) متغیرهای x, y, z و result از نوع int اعلان شوند (در عبارات مجزا).

(c) به کاربر اعلان شود تا سه عدد صحیح وارد برنامه سازد.

(d) سه مقدار صحیح از صفحه کلید دریافت آنها را در متغیرهای x, y و z ذخیره کند.

(e) حاصلضرب سه مقدار موجود در متغیرهای x, y و z را بدست آورده و آنرا در متغیر result ذخیره کند.



## 36 فصل دوم \_\_\_\_\_ مقدمه ای بر برنامه نویسی C++

(f) عبارت "The product is" را قبل از مقدار متغیر **result** چاپ کند.

(g) مقداری از **main** باز گرداند، تا نشان دهد برنامه با موفقیت پایان پذیرفته است.

۲-۵ با استفاده از عبارات نوشته شده در تمرین ۴-۲، یک برنامه کامل بنویسید که حاصلضرب سه مقدار صحیح را بدست آورده و نتیجه را به نمایش در آورد. [توجه: از عبارات **using** استفاده کنید.]

۲-۶ خطاهای موجود در عبارات زیر را تشخیص داده و آنها را اصلاح کنید (فرض کنید که از عبارت **using** **std::cout** استفاده شده است):

- a) `if ( c < 7 ) ;  
cout << "c is less than 7\n";`
- b) `if ( c => 7 )  
cout << "c is equal to or greater than 7\n";`

### پاسخ خودآزمایی

۲-۱ (a) **main** (b) براکت چپ، براکت راست (c) سیمکولن (d) خط جدید. **if** (e)

۲-۲ (a) اشتباه. توضیحات هیچ عملی به هنگام اجرای برنامه انجام نمی دهند. از آنها فقط برای مستند کردن و افزایش خوانائی برنامه استفاده می شود.

(b) صحیح.

(c) صحیح.

(d) صحیح.

(e) اشتباه. زبان C++ حساس به موضوع است، از اینرو متغیرها با یکدیگر فرق دارند.

(f) صحیح.

(g) صحیح.

(h) اشتباه. عملگرهای /, \* و % دارای اولویت یکسان بوده و عملگرهای + و - از اولویت پایین تر برخوردار هستند.

(i) اشتباه. یک عبارت خروجی با استفاده از **cout** حاوی چندین توالی **\n** می تواند در چندین خط چاپ شود.

۲-۳

- a) `int c, thisIsVariable, q76354, number;`
- b) `std::cout << "Enter an integer: " ;`
- c) `std::cin >> age;`
- d) `if ( number != 7 )  
std::cout << "The variable number is not equal to 7 \n";`
- e) `std::cout << "This is a C++ program\n";`
- f) `std::cout << "This is a C++\nprogram\n";`
- g) `std::cout << "This\nis\na\nC++\nprogram\n";`
- h) `std::cout << "This \tis\ta\tC++\tprogram\n";`

۲-۴

- a) `// Calculate the product of three integers`
- b) `int x;`



مقدمه ای بر برنامه نویسی C++ فصل دوم 37

```

int y;
int z;
int result;
c) cout << "Enter three integers: ";
d) cin >> x >> y >> z;
e) result = x * y * z;
f) cout << "The product is " << result << endl;
g) return 0;

```

۲-۵

```

1 // Calculate the product of three integers
2 #include <iostream>
3
4 using std::cout;
5 using std::cin;
6 using std::endl;
7
8 //function main begins program execution
9 int main()
10 {
11     int x; //first integer to multiply
12     int y; //second integer to multiply
13     int z; //third integer to multiply
14     int result; //the product of the three integer
15
16     cout << "Enter three integers: "; //prompt user for data
17     cin >> x >> y >> z; //read three integers from user
18     result = x * y * z; //multiply the three integers; store result
19     cout << "The product is " << result << endl; //print result; end line
20
21     return 0; //indicate program executed successfully
22 } // end function main

```

۲-۶ (a) خطا. سیمکولن پس از پرانتز در سمت راست قرار گرفته است. برای اصلاح این خطا بایستی سیمکولن پس از پرانتز سمت راست را حذف کنید. [توجه: در نتیجه اجرای این قسمت از برنامه با خطای موجود، عبارت پس از شرط if در هر حالتی به اجرا در خواهد آمد.]

(b) خطا: عملگر رابطه‌ای بصورت >= تایپ شده است. برای اصلاح این خطا، عملگر باید بفرم >= تایپ شود.

### تمرینات

۲-۷ در ارتباط با هر کدامیک از شی‌های زیر توضیح دهید:

```

std::cin (a)
std::cout (b)

```

۲-۸ جاهای خالی در عبارات زیر را با کلمات مناسب پر کنید:

(a) ..... در افزایش خوانائی و مستند کردن یک برنامه نقش دارند.

(b) از شی ..... برای چاپ اطلاعات بر روی صفحه نمایش استفاده می‌شود.

(c) عبارت ..... در تصمیم‌گیری بکار گرفته می‌شود.

(d) معمولا محاسبات توسط عبارات ..... انجام می‌شوند.

(e) شی ..... مقادیر را از صفحه کلید دریافت می‌کند.



### 38 فصل دوم \_\_\_\_\_ مقدمه‌ای بر برنامه‌نویسی C++

۹-۲ یک عبارت تک جمله‌ای در C++ بنویسید که موارد مورد تقاضا را برآورده سازد:

(a) جمله "Enter two numbers" چاپ شود.

(b) حاصلضرب متغیرهای **b** و **c** در متغیر **a** ذخیره شود.

(c) مشخص کنید که برنامه عمل محاسبه پرداخت دستمزد را انجام می‌دهد (عبارت توضیحی بنویسید).

(d) سه مقدار صحیح از صفحه کلید دریافت و در متغیرهای صحیح **a**, **b** و **c** قرار دهد.

۱۰-۲ کدام عبارات صحیح و کدام اشتباه است. اگر عبارتی اشتباه است علت آنرا توضیح دهید.

(a) ارزیابی عملگرهای C++ از سمت چپ به راست صورت می‌گیرد.

(b) تمام اسامی متغیر زیر معتبر هستند:

`_under_bar_ , m928134, t5, j7, her_sales, his_account_total, a, b, c, z, z2`

(c) عبارت `"a = 5;"`، مثالی از یک عبارت تخصیصی است.

(d) یک عبارت محاسباتی معتبر C++ بدون وجود پرانتز از سمت چپ به راست ارزیابی می‌شود.

(e) تمام اسامی متغیر زیر معتبر نمی‌باشند:

`3g, 87, 67h2, h22, 2h`

۱۱-۲ جاهای خالی در عبارات زیر را با کلمات مناسب پر کنید:

(a) کدام عملیات محاسباتی در سطح یکسانی از تقدم ضرب قرار دارند؟.....

(b) زمانیکه پرانتزها بصورت تودرتو هستند، کدام جفت پرانتز ابتدا ارزیابی می‌شود؟.....

(c) مکانی در حافظه کامپیوتر که می‌تواند در هر بار مقدار متفاوتی داشته باشد، ..... نامیده می‌شود.

۱۲-۲ با انجام عبارات زیر چه اتفاقی رخ می‌دهد. با فرض  $x = 2$  و  $y = 3$

- a) `cout << x;`
- b) `cout << x + x;`
- c) `cout << "x=";`
- d) `cout << "x = " << x;`
- e) `cout << x + y << " = " << y + x;`
- f) `z = x + y;`
- g) `cin >> x >> y;`
- h) `//cout << "x + y = " << x + y;`
- i) `cout << "\n";`

۱۳-۲ کدام یک از عبارات C++ زیر حاوی متغیرهای هستند که مقادیر آنها تغییر خواهد یافت؟

- a) `cin >> b >> c >> d >> e >> f;`
- b) `p = i + j + k + 7;`
- c) `cout << "variables whose values are replaced"`
- d) `cout << "a = 5";`

۱۴-۲ با توجه به معادله  $y = ax^3 + 7$  کدامیک از عبارات زیر پاسخ صحیح این معادله هستند؟

- a)  $y = a * x * x * x + 7;$
- b)  $y = a * x * x * (x + 7);$
- c)  $y = (a * x) * x * (x + 7);$
- d)  $y = (a * x) * x * x + 7;$
- e)  $y = a * (x * x * x) + 7;$



مقدمه ای بر برنامه نویسی C++ فصل دوم 39

f)  $y = a * x * (x * x + 7);$

۲-۱۵ ترتیب ارزیابی عملگرهای زیر را در عبارات C++ مشخص سازید و مقدار x را بدست آورید.

a)  $x = 7 + 3 * 6 / 2 - 1;$

b)  $x = 2 \% 2 + 2 * 2 - 2 / 2;$

c)  $x = (3 * 9 * (3 + (9 * 3 / (3))));$

۲-۱۶ برنامه ای بنویسید که دو عدد از کاربر دریافت و سپس مجموع، ضرب، تفریق و تقسیم آنها را بدست آورده و چاپ کند.

۲-۱۷ برنامه ای بنویسید که اعداد 1 تا 4 را بر روی یک خط چاپ کند، به نحوی که مابین هر جفت یک کاراکتر فاصله

قرار گرفته باشد. برنامه را با استفاده از روش های زیر بنویسید:

(a) با استفاده از یک عبارت خروجی با یک عملگر درج.

(b) با استفاده از یک عبارت خروجی با چهار عملگر درج.

(c) با استفاده از چهار عبارت.

۲-۱۸ برنامه ای بنویسید که از کاربر تقاضای دریافت دو عدد صحیح کرده و آنها را دریافت و سپس عدد بزرگتر را قبل از

جمله "is larger" چاپ کند. اگر دو عدد با هم برابر باشند، پیغام "These numbers are equal" چاپ شود.

۲-۱۹ برنامه ای بنویسید که از طریق صفحه کلید سه عدد صحیح دریافت کرده و سپس مجموع، میانگین، حاصلضرب،

کوچکترین و بزرگترین عدد را چاپ کند. خروجی بایستی بفرم زیر طراحی شود:

```
Input three different integers: 13 27 14
Sum is 54
Average is 18
Product is 4914
Smallest is 13
Largest is 27
```

۲-۲۰ برنامه ای بنویسید که از طریق کاربر مقدار شعاع یک دایره را دریافت و میزان قطر دایره، محیط و مساحت

آنها نمایش درآورد. از فرمول های زیر استفاده کنید (r شعاع دایره است):

$$2r = \text{قطر}, \quad 2\pi r = \text{محیط دایره}, \quad \pi r^2 = \text{مساحت}.$$

۲-۲۱ برنامه ای بنویسید که بتواند یک مستطیل، یک لوزی، یک فلش و یک لوزی را همانند الگوهای زیر با

استفاده از کاراکتر \* ترسیم کند:

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

۲-۲۲ کد زیر چه عبارتی چاپ می کند؟

```
cout << "\n**\n***\n****\n*****\n"<<endl;
```



## 40 فصل دوم \_\_\_\_\_ مقدمه‌ای بر برنامه‌نویسی C++

۲-۲۳ برنامه‌ای بنویسید که پنج عدد در یافت و بزرگترین و کوچکترین آنها را چاپ کند. از تکنیک‌های معرفی شده در این فصل استفاده کنید.

۲-۲۴ برنامه‌ای بنویسید که یک عدد صحیح دریافت و تعیین نماید که آیا آن عدد زوج است یا فرد؟

۲-۲۵ برنامه‌ای بنویسید که دو عدد صحیح دریافت و تعیین نماید که آیا عدد اولی حاصلضربی از عدد دوم است یا خیر؟

۲-۲۶ با استفاده از هشت عبارت خروجی، الگوی زیر را تولید و سپس همین الگو را در صورت امکان با استفاده از چهار عبارت خروجی پیاده‌سازی نمایید.

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

۲-۲۷ در این فصل با اعداد صحیح و نوع `int` آشنا شدید. با این همه `C++` قادر به نمایش حروف کوچک، بزرگ و انواع متنوعی از نمادهای خاص نیز است. زبان `C++` با استفاده از مقادیر داخلی صحیح و کوچک مبادرت به نمایش کاراکترها می‌کند. مجموعه کاراکترهای بکار رفته توسط یک کامپیوتر و مقادیر صحیح متناظر با آن کاراکترها، مجموعه کاراکتری کامپیوتر نامیده می‌شود. می‌توان یک کاراکتر را با قرار دادن آن کاراکتر مابین یک کوتیشن (`single quotes`) چاپ کرد. برای مثال:

```
cout << 'A'; // print an uppercase A
```

می‌توان معادل صحیح یک کاراکتر را با استفاده از `static_cast` و بفرم زیر چاپ کرد:

```
cout <<static_cast< int >( 'A' ); //print 'A' as an integer
```

که به اینحالت عملیات `cast` گفته می‌شود (در فصل چهارم به معرفی جامعتر `cast` خواهیم پرداخت). هنگامی که عبارت فوق اجرا گردد، مقدار 65 را چاپ خواهد کرد (بر روی سیستمی که از مجموعه کاراکتری ASCII استفاده می‌کند).

برنامه‌ای بنویسید که معادل صحیح تعدادی از حروف بزرگ، کوچک، ارقام و نمادهای ویژه را چاپ کند.

۲-۲۸ برنامه‌ای بنویسید که یک عدد متشکل از پنج رقم از کاربر دریافت کرده و سپس آن عدد را به ارقام معجزا از هم تبدیل و هر یک را با سه فاصله از رقم بعدی به چاپ رساند. برای مثال اگر کاربر عدد 42339 را وارد کند، خروجی برنامه باید:

```
4 2 3 3 9
```

باشد. از پنجره دستور برای دریافت ورودی و نمایش خروجی استفاده کنید [این تمرین با استفاده از تکنیک‌های گفته شده در این فصل قابل اجرا است. نیاز است تا از عملیات تقسیم و باقیمانده برای جدا کردن هر رقم استفاده کنید].

۲-۲۹ فقط با استفاده از تکنیک‌های برنامه‌نویسی معرفی شده در این فصل، برنامه‌ای بنویسید که مربع و مکعب اعداد از 0 تا 5 را محاسبه و نتایج آنرا در جدولی بصورت زیر به نمایش درآورد:





مقدمه ای بر برنامه نویسی C++ \_\_\_\_\_ فصل دوم 41

integer	square	cube
0	0	0
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125

برای نمایش و دریافت ورودی از پنجره دستور استفاده کنید [نکته: این برنامه نیازی به دریافت ورودی از سوی کاربر ندارد].