فصل سوم

«انتخاب»

همهٔ برنامه‌هایی که در دو فصل اول بیان شد، به شکل ترتیبی اجرا می‌شوند، یعنی دستورات برنامه به ترتیب از بالا به پایین و هر کدام دقیقا یک بار اجرا می‌شوند. در این فصل نشان داده می‌شود چگونه از دستورالعمل‌های انتخاب[1] جهت انعطاف‌پذیری بیشتر برنامه استفاده کنیم. همچنین در این فصل انواع صحیح که در C++ وجود دارد بیشتر بررسی می‌گردد.

3-1 دستور if

دستور **if** موجب می‌شود برنامه به شکل شرطی اجرا شود. نحو آن به گونهٔ زیر است:

```
If (condition) statement;
```

Condition که **شرط** نامیده می‌شود یک عبارت صحیح است (عبارتی که با یک مقدار صحیح برآورد می‌شود) و statement می‌تواند هر فرمان قابل اجرا باشد. Statement وقتی اجرا خواهد شد که *condition* مقدار غیر صفر داشته باشد.

---

1 – Selection

دقت کنید که شرط باید درون پرانتز قرار داده شود.

× مثال 1-3 آزمون بخش‌پذیری

این برنامه بررسی می‌کند که یک عدد صحیح مثبت بر عدد دیگر قابل تقسیم نباشد:

```
int main()
{  int n, d;
   cout << "Enter two positive integers: ";
   cin >> n >> d;
   if (n%d) cout << n << " is not divisible by "
                 << d << endl;
}
```

در اولین اجرا، اعداد 66 و 7 را وارد می‌کنیم:

```
Enter two positive integers: 66 7
66 is not divisible by 7
```

مقدار 66%7 برابر با 3 برآورد می‌گردد. چون این مقدار، یک عدد صحیح غیرصفر است، پس شرط به عنوان درست تفسیر می‌شود و در نتیجه دستور cout اجرا شده و پیغام عدم قابلیت تقسیم چاپ می‌شود.

برنامهٔ بالا را دوباره اجرا می‌نماییم و این دفعه اعداد 56 و 7 را وارد می‌کنیم:

```
Enter two positive integers: 56 7
```

مقدار 56%7 برابر با 0 برآورد می‌شود که این به معنی نادرست تفسیر می‌گردد، پس دستور cout نادیده گرفته شده و هیچ پیغامی روی صفحه چاپ نمی‌شود.

در C++ هر وقت یک عبارت صحیح به عنوان یک شرط استفاده شود، مقدار 0 به معنی «نادرست» و همهٔ مقادیر دیگر به معنی «درست» است.

برنامهٔ مثال 1-3 ناقص به نظر می‌آید زیرا اگر n بر d قابل تقسیم باشد، برنامه هیچ عکس‌العملی نشان نمی‌دهد. این نقص به کمک دستور if..else رفع می‌شود.

3-2 دستور if..else

دستور if..else موجب می‌شود بسته به این که شرط درست باشد یا خیر، یکی از دو دستورالعمل فرعی اجرا گردد. نحو این دستور به شکل زیر است:

```
if (condition) statement1;
else statement2;
```

condition همان شرط مساله است که یک عبارت صحیح می‌باشد و statement1 و statement2 فرمان‌های قابل اجرا هستند. اگر مقدار شرط، غیر صفر باشد، statement1 اجرا خواهد شد وگرنه statement2 اجرا می‌شود.

× مثال 2-3 یک آزمون دیگر قابلیت تقسیم

این برنامه مانند برنامهٔ مثال 1-3 است بجز این که دستور if با دستور if..else جایگزین شده است:

```
int main()
{  int n, d;
   cout << " Enter two positive integers: ";
   cin >> n >> d;
   if (n%d) cout << n << " is not divisible by "
                 << d << endl;
   else cout << n << " is divisible by " << d << endl;
}
```

حالا وقتی در این برنامه اعداد 56 و 7 را وارد کنیم، برنامه پاسخ می‌دهد که 56 بر 7 قابل تقسیم است:

```
Enter two positive integers: 56 7
56 is divisible by 7
```

چون حاصل 56%7 برابر با صفر است، پس این عبارت به عنوان نادرست تفسیر می‌گردد. در نتیجه دستور بعد از if نادیده گرفته شده و دستور بعد از else اجرا می‌شود. توجه کنید که if..else به تنهایی یک دستور است، گر چه به دو سمیکولن نیاز دارد.

3-4 عملگرهای مقایسه‌ای

در C++ شش عملگر مقایسه‌ای وجود دارد: < و > و <= و >= و == و =! . هر یک از این شش عملگر به شکل زیر به کار می‌روند:

```
x < y   // x کوچک‌تر از y است
x > y   // x بزرگ‌تر از y است
x <= y  // x کوچک‌تر یا مساوی y است
x >= y  // x بزرگ‌تر یا مساوی y است
x == y  // x مساوی با y است
x != y  // x مساوی با y نیست
```

این‌ها می‌توانند برای مقایسهٔ مقدار عبارات با هر نوع ترتیبی استفاده شوند. عبارت حاصل به عنوان یک شرط تفسیر می‌شود. مقدار این شرط صفر است اگر شرط نادرست باشد و غیر صفر است اگر شرط درست باشد. برای نمونه، عبارت 7×8<6×9 برابر با صفر ارزیابی می‌شود، به این معنی که این شرط نادرست است.

× مثال 3-3 کمینهٔ دو عدد صحیح

این برنامه مشخص می‌کند که از دو عدد صحیح ورودی، کدام یک کوچک‌تر است:

```
int main()
{  int m, n;
   cout << "Enter two integers: ";
   cin >> m >> n;
   if ( m < n ) cout << m << " is the minimum." << endl;
   else cout << n << " is the minimum." << endl;
}
```

```
Enter two integers: 77 55
55 is the minimum.
```

دقت کنید که در C++ عملگر جایگزینی با عملگر برابری فرق دارد. عملگر جایگزینی یک مساوی تکی " = " است ولی عملگر برابری، دو مساوی " = = " است. مثلا

دستور x = 33; مقدار 33 را در x قرار می‌دهد ولی دستور x == 33; بررسی می‌کند که آیا مقدار x با 33 برابر است یا خیر. درک این تفاوت اهمیت زیادی دارد.

```
x = 33;        // مقدار33 را به X تخصیص می‌دهد
x == 33;       // صفر ارزیابی می‌شود (به معنی نادرست) مگر این که مقدار x برابر با 33 باشد
```

× مثال 4-3 یک خطای برنامه‌نویسی متداول

این برنامه خطادار است:

```
int main()
{  int n;
   cout << "Enter an integer: ";
   cin >> n;
   if (n = 22) cout << "n = 22" << endl;  // LOGICAL ERROR!
   else cout << "n != 22" << endl;
}
```

```
Enter an integer: 77
n = 22
```

ظاهرا منطق برنامهٔ فوق به این گونه است که عددی از ورودی دریافت می‌شود و اگر این عدد با 22 برابر بود، پیغام برابری چاپ می‌شود و در غیر این صورت پیغام عدم برابری چاپ می‌گردد. ولی اجرای بالا نشان می‌دهد که برنامه درست کار نمی‌کند. عدد 77 وارد شده ولی پیغام n = 22 در خروجی چاپ شده است! ایراد در خط پنجم برنامه است. عبارت n = 22 مقدار 22 را در n قرار داده و مقدار قبلی آن که 77 است را تغییر می‌دهد. اما عبارت n = 22 به عنوان شرط دستور if استفاده شده پس به عنوان یک عبارت صحیح با مقدار 22 برآورد می‌شود. لذا شرط (n = 22) به عنوان «درست» تفسیر می‌شود زیرا فقط مقدار 0 به معنای «نادرست» است. به همین دلیل دستور قبل از else اجرا می‌شود. خط پنجم باید این‌طور نوشته می‌شد:

```
if (n = = 22) cout << "n = 22" << endl;        // CORRECT
```

خطای نشان داده شده در این مثال، ***خطای منطقی***[1] نام دارد. این نوع خطا، بدترین نوع خطاهاست. خطاهای زمان کامپایل (مانند از قلم افتادن یک سمیکولن) به

1 – Logical Error

وسیلهٔ کامپایلر گرفته می‌شود. خطاهای زمان اجرا (مانند تقسیم بر صفر) نیز به وسیلهٔ سیستم عامل گرفته می‌شود اما خطای منطقی را نمی‌توان با این ابزارها کشف کرد.

× مثال 5-3 کمینهٔ سه عدد صحیح

این مثال شبیه مثال 3-3 است با این تفاوت که از سه عدد صحیح استفاده می‌کند:

```
int main()
{  int n1, n2, n3;
   cout << "Enter three integers: ";
   cin >> n1 >> n2 >> n3;
   int min=n1;              // now min <= n1
   if (n2 < min) min = n2;  // now min <= n1 and  n2
   if (n3 < min) min = n3;  // now min <= n1 ,n2 ,and n3
   cout << "Their minimum is " << min << endl;
}
```

```
Enter three integers: 77 33 55
Their minimum is 33
```

سه توضیح ذکر شده در برنامه، نحوهٔ پیشرفت کار را نشان می‌دهد: ابتدا min برابر n1 فرض می‌شود، لذا min کمینهٔ مجموعهٔ {n1} می‌شود. پس از اجرای اولین if، مقدار min برابر با n2 می‌شود اگر n2 از مقدار فعلی min کوچک‌تر باشد. پس min برابر کمینهٔ مجموعهٔ {n1,n2} می‌شود. آخرین دستور if، مقدار min را برابر با n3 قرار می‌دهد اگر n3 از مقدار فعلی min کوچک‌تر باشد. بنابراین در نهایت مقدار min برابر با کمینهٔ مجموعهٔ {n1, n2, n3} خواهد شد.

3-5 بلوک‌های دستورالعمل

یک ***بلوک دستورالعمل*** زنجیره‌ای از دستورالعمل‌هاست که درون براکت {} محصور شده، مانند این:

```
{  int temp=x;
   x = y;
   y = temp;
}
```

در برنامه‌های C++ یک بلوک دستورالعمل مانند یک دستورالعمل تکی است. یعنی هر جا که یک دستورالعمل تنها بتواند استفاده شود، یک بلوک دستورالعمل نیز می‌تواند استفاده شود. به مثال بعدی توجه کنید.

× مثال 6-3 یک بلوک دستورالعمل درون یک دستور if

این برنامه دو عدد صحیح را گرفته و به ترتیب بزرگ‌تری، آن‌ها را چاپ می‌کند:

```
int main()
{  int x, y;
   cout << "Enter two integers: ";
   cin >> x >> y;
   if (x > y)  { int temp = x;
                 x = y;
                 y = temp;
               }                          //swap x and y
   cout << x << " <= " << y << endl;
}
```

```
Enter two integers: 66 44
44 <= 66
```

سه دستور درون بلوک، مقادیر x و y را به ترتیب بزرگ‌تری مرتب می‌کنند بدین شکل که اگر آن‌ها خارج از ترتیب باشند، جای آن دو را عوض می‌کنند. برای این جابجایی به سه گام متوالی و یک محل ذخیره‌سازی موقتی احتیاج داریم که در این‌جا temp نامیده شده. برنامه یا باید هر سه دستورالعمل را اجرا کند و یا هیچ یک را نباید اجرا کند. وقتی این سه دستور را درون بلوک دستورالعمل قرار دهیم، منظور فوق برآورده می‌شود. توجه کنید که متغیر temp درون بلوک تعریف شده است. این سبب می‌شود که متغیر مذکور درون بلوک، یک **متغیر محلی**[1] باشد. یعنی این متغیر فقط وقتی ایجاد می‌شود که بلوک اجرا شود. اگر شرط نادرست باشد (یعنی x<=y باشد) متغیر temp هرگز موجود نخواهد شد. این مثال، روش مناسبی برای محلی کردن اشیا را نشان می‌دهد، طوری که اشیا وقتی ایجاد می‌شوند که به آن‌ها نیاز است. همچنین توجه کنید که یک برنامهٔ C++ خودش یک بلوک دستورالعمل است که توسط تابع

1 – Local Variable

اصلی main() ساخته شده است. یادآوری می‌کنیم که حوزهٔ متغیر، قسمتی از یک برنامه است که متغیر می‌تواند در آن استفاده شود(بخش 5-1). این حوزه، از نقطه‌ای که متغیر اعلان می‌شود شروع شده و تا پایان همان بلوک ادامه می‌یابد. پس یک بلوک می‌تواند به عنوان محدودهٔ حوزهٔ متغیر استفاده شود. یکی از نتایج مهم این کار آن است که می‌توانیم از متغیرهای متفاوتی با یک نام در قسمت‌های مختلف برنامه استفاده کنیم.

× مثال 7-3 استفاده از بلوک‌ها به عنوان محدودهٔ حوزه

در این برنامه سه متغیر مختلف با نام n استفاده شده است:

```
int main()
{  int n=44;
   cout << "n = " << n << endl;
   {  int n;                          // scope extends over 4 lines
      cout << "Enter an integer: ";
      cin >> n;
      cout << "n = " << n << endl;
   }
   {  cout << " n = " << n << endl; // n that was declared first
   }
   {  int n;                          // scope extends over 2 lines
      cout << "n = " << n << endl;
   }
   cout << "n = " << n << endl;     // n that was declared first
}
```

```
n = 44
Enter an integer: 77
n = 77
n = 44
n = 4251897
n = 44
```

برنامهٔ بالا سه بلوک داخلی دارد. اولین بلوک یک n جدید اعلان می‌کند که فقط درون همان بلوک، معتبر و موجود است. این n متغیر n اصلی را پنهان می‌کند. بنابراین وقتی مقدار 77 در این بلوک از ورودی دریافت می‌شود، این مقدار درون n محلی قرار می‌گیرد و مقدار n اصلی بدون تغییر می‌ماند. در دومین بلوک n جدیدی تعریف

نمی‌شود، لذا حوزهٔ n اصلی این بلوک را نیز شامل می‌شود. پس در سومین دستور خروجی، مقدار n اصلی یعنی 44 چاپ می‌شود. بلوک سوم برنامه نیز مانند بلوک اول یک n جدید تعریف می‌کند که n اصلی را پنهان می‌نماید، اما این n جدید مقداردهی نمی‌شود. بنابراین در چهارمین خروجی، یک مقدار زباله چاپ می‌شود. در خط انتهایی برنامه، تمام بلوک‌های محلی به پایان می‌رسند. به همین خاطر وقتی در این خط دستور چاپ برای n صادر می‌شود، مقدار n اصلی یعنی 44 چاپ می‌شود.

3-6 شرط‌های مرکب

شرط‌هایی مانند n%d و x>=y می‌توانند به صورت یک شرط مرکب با هم ترکیب شوند. این کار با استفاده ازعملگرهای منطقی && (and) و || (or) و ! (not) صورت می‌پذیرد. این عملگرها به شکل زیر تعریف می‌شوند:

p && q درست است اگر و تنها اگر هم p و هم q هر دو درست باشند

p || q نادرست است اگر و تنها اگر هم p و هم q هر دو نادرست باشند

p! درست است اگر و تنها اگر p نادرست باشد

برای مثال (n%d || x>=y) نادرست است اگر و تنها اگر n%d برابر صفر و x کوچک‌تر از y باشد.

سه عملگر منطقی بالا معمولا با استفاده از ***جداول درستی*** به گونهٔ زیر بیان می‌شوند:

p	q	P&&q
T	T	T
T	F	F
F	T	F
F	F	F

p	q	p||q
T	T	T
T	F	T
F	T	T
F	F	F

p	!p
T	F
F	T

طبق جدول‌های فوق اگر p درست و q نادرست باشد، عبارت p&&q نادرست و عبارت p||q درست است.

مثال بعدی همان مسألهٔ مثال 5-3 را حل می‌کند ولی این کار را با استفاده از شرط‌های مرکب انجام می‌دهد.

× مثال 8-3 استفاده از شرط‌های مرکب

برنامهٔ زیر مانند برنامهٔ مثال 5-3 است. این نسخه برای یافتن کمینهٔ سه عدد از شرط‌های مرکب استفاده کرده است:

```
int main()
{  int n1, n2, n3;
   cout << "Enter three integers: ";
   cin >> n1 >> n2 >> n3;
   if (n1<=n2 && n1<=n3) cout << "Their minimum is "
       << n1 <<endl;
   if (n2<=n1 && n2<=n3) cout << "Their minimum is "
       << n2 <<endl;
   if (n3<=n1 && n3<=n2) cout << "Their minimum is "
       << n3 <<endl;
}
```

```
Enter three integers: 77 33 55
Their minimum is 33
```

برنامه‌ای که در این مثال آمد هیچ مزیتی بر مثال 5-3 ندارد و فقط نحوهٔ استفاده از شرط‌های مرکب را بیان می‌کند. در مثال زیر هم از شرط مرکب استفاده شده است.

× مثال 9-3 ورودی کاربر پسند

این برنامه به کاربر امکان می‌دهد که برای پاسخ مثبت "y" یا "Y" را وارد کند:

```
int main()
{  char ans;
   cout << "Are you enrolled (y/n): ";
   cin >> ans;
   if (ans= ='Y' || ans= ='y') cout << "You are enrolled.\n";
   else cout << "You are not enrolled.\n";
}
```

```
Are you enrolled (y/n): N
```

```
You are not enrolled.
```

برنامهٔ بالا از کاربر پاسخی می‌خواهد و y و n را به عنوان جواب‌های ممکن پیشنهاد می‌دهد. اما هر کاراکتر دیگری را هم می‌پذیرد و اگر آن کاراکتر 'y' یا 'Y' نباشد، فرض می‌کند که پاسخ کاربر "no" است.

3-7 ارزیابی میانبری

عملگرهای && و || به دو عملوند نیاز دارند. یعنی به دو مقدار نیاز دارند تا مقایسه را روی آن دو انجام دهند. شرط‌های مرکب که از && و || استفاده می‌کنند عملوند دوم را بررسی نمی‌کنند مگر این که لازم باشد. جداول درستی نشان می‌دهد که p&&q نادرست است اگر p نادرست باشد. در این حالت دیگر نیازی نیست که q بررسی شود. همچنین p||q درست است اگر p درست باشد و در این حالت هم نیازی نیست که q بررسی شود. در هر دو حالت گفته شده، با ارزیابی عملوند اول به سرعت نتیجه معلوم می‌شود. این کار ***ارزیابی میانبری*** نامیده می‌شود.

× مثال 10-3 ارزیابی میانبری

برنامهٔ زیر بخش‌پذیری اعداد صحیح را بررسی می‌کند:

```
int main()
{  int n, d;
   cout << "Enter two positive integers: ";
   cin >> n >> d;
   if (d != 0 && n%d == 0) cout << d << " divides " << n
                                << endl;
   else cout << d << " does not divide " << n << endl;
}
```

در اجرای زیر، d مثبت و n%d صفر است. بنابراین شرط مرکب درست است:

```
Enter two integers: 300 5
5 divides 300
```

در اجرای بعدی، d مثبت است اما n%d صفر نیست. بنابراین شرط مرکب نادرست است:

```
Enter two integers: 300 7
7 does not divide 300
```

در آخرین اجرا، d صفر است. پس به سرعت برآورد می‌شود که شرط مرکب نادرست است بدون این که عبارت دوم یعنی n%d == 0 ارزیابی شود:

```
Enter two integers: 300 0
0 does not divide 300
```

ارزیابی میانبری در مثال بالا از خرابی برنامه جلوگیری می‌کند زیرا وقتی d صفر است، رایانه نمی‌تواند عبارت n%d را محاسبه کند.

3-8 عبارات منطقی

یک ***عبارت منطقی*** شرطی است که یا درست است یا نادرست. در مثال قبلی عبارات d>0 و n%d == 0 و (d>0 && n%d==0) عبارات منطقی هستند. قبلا دیدیم که عبارات منطقی با مقادیر صحیح ارزیابی می‌شوند. مقدار صفر به معنای نادرست و هر مقدار غیر صفر به معنای درست است. به عبارات منطقی «عبارات بولی» هم می‌گویند.

چون همهٔ مقادیر صحیح ناصفر به معنای درست تفسیر می‌شوند، عبارات منطقی اغلب تغییر قیافه می‌دهند. برای مثال دستور

```
if (n) cout << "n is not zero";
```

وقتی n غیر صفر است عبارت n is not zero را چاپ می‌کند زیرا عبارت منطقی (n) وقتی مقدار n غیر صفر است به عنوان درست تفسیر می‌گردد. کد زیر را نگاه کنید:

```
if (n%d) cout << "n is not a multiple of d";
```

دستور خروجی فقط وقتی که n%d ناصفر است اجرا می‌گردد و n%d وقتی ناصفر است که n بر d بخش‌پذیر نباشد. گاهی ممکن است فراموش کنیم که عبارات منطقی مقادیر صحیح دارند و این فراموشی باعث ایجاد نتایج غیر منتظره و نامتعارف شود.

× مثال 3-11 یک خطای منطقی دیگر

این برنامه خطادار است:

```
int main()
{  int n1, n2, n3;
   cout << "Enter three integers: ";
   cin >> n1 >> n2 >> n3;
   if (n1 >= n2 >= n3) cout << "max = " << n1;  // LOGICAL ERROR!
}
```

```
Enter three integers: 0 0 1
max = 0
```

منشأ خطا در برنامهٔ بالا این اصل است که عبارات منطقی مقدارهای عددی دارند. چون عبارت (n1>=n2>=n3) از چپ به راست ارزیابی می‌شود، به ازای ورودی‌های فوق اولین بخش ارزیابی یعنی n1>=n2 درست است چون 0>=0 اما «**درست**» به شکل عدد 1 در حافظه نگهداری می‌شود. سپس این مقدار با مقدار n3 که 1 می‌باشد مقایسه می‌شود. یعنی عبارت 1>=1 ارزیابی می‌شود که این هم درست است. نتیجه این است که کل عبارت به عنوان درست تفسیر می‌شود گرچه در حقیقت این طور نیست! (0 بیشینهٔ 0 و 1 نیست)

ایراد کار این‌جاست که خط اشتباه به طور نحوی صحیح است. بنابراین نه کامپایلر می‌تواند خطا بگیرد و نه سیستم‌عامل. این نوع دیگری از خطای منطقی است که با آن‌چه در مثال 3-4 مطرح شد قابل مقایسه است. نتیجهٔ این مثال آن است که: «همیشه به خاطر داشته باشید عبارات منطقی مقدار عددی دارند، بنابراین شرط‌های مرکب می‌توانند گول‌زننده باشند».

3-9 دستورهای انتخاب تودرتو

دستورهای انتخاب می‌توانند مانند دستورالعمل‌های مرکب به کار روند. به این صورت که یک دستور انتخاب می‌تواند درون دستور انتخاب دیگر استفاده شود. به این روش، ***جملات تودرتو*** می‌گویند.

× مثال 12-3 دستورهای انتخاب تودرتو

این برنامه همان اثر مثال 10-3 را دارد:

```
int main()
{  int n, d;
   cout << "Enter two positive integers: ";
   cin >> n >> d;
   if (d != 0)
      if (n%d = = 0) cout << d << " divides " << n << endl;
      else cout << d << " does not divide " << n << endl;
   else cout << d << " does not divide " << n << endl;
}
```

در برنامهٔ بالا، دستور if..else دوم درون دستور if..else اول قرار گرفته است. پس if..else دوم وقتی اجرا می‌شود که d صفر نباشد. توجه کنید که در این‌جا مجبوریم دو بار از عبارت does not divide استفاده کنیم. اولی در اولین دستور if..else قرار گرفته و زمانی اجرا می‌شود که d صفر نباشد و n%d صفر گردد. دومی هم وقتی اجرا می‌شود که d صفر باشد.

وقتی دستور **if..else** به شکل تو در تو به کار می‌رود، کامپایلر از قانون زیر جهت تجزیه این دستورالعمل مرکب استفاده می‌کند:

« هر **else** با آخرین **if** تنها جفت می‌شود.»

با به‌کارگیری این قانون، کامپایلر به راحتی می‌تواند کد پیچیدهٔ زیر را رمز گشایی کند:

```
if (a > 0) if (b > 0) ++a; else if (c > 0)        //BAD CODING STYLE
if (a > 4) ++b; else  if (b < 4) ++c; else -a;    //BAD CODING STYLE
else if (c < 4) --b; else --c; else a = 0;        //BAD CODING STYLE
```

برای این که کد بالا را خواناتر و قابل فهم کنیم، می‌توانیم آن را به شکل زیر بنویسیم:

```
if (a > 0)
   if (b > 0) ++a;
   else
      if (c > 0)
         if (a < 4) ++b;
```

```
         else
            if (b < 4) ++c;
            else –a;
      else
         if (c < 4) –b;
         else –c;
else a = 0;
```

یا به این شکل :

```
if (a > 0)
   if (b > 0) ++a;
   else if (c > 0)
      if (a < 4) ++b;
      else if (b < 4) ++c;
      else –a;
   else if (c < 4) –b;
   else –c;
else a = 0;
```

در شیوهٔ دوم عبارات else if زیرِ هم و در یک راستا نوشته می‌شوند.

× مثال 13–3 استفاده از دستورهای انتخاب تودرتو

این برنامه همان اثر مثال‌های 5–3 و 8–3 را دارد. در این نسخه برای یافتن کمینهٔ سه عدد صحیح از دستورهای if..else تودرتو استفاده می‌شود:

```
int main()
{  int n1, n2, n3;
   cout << "Enter three integers: ";
   cin >> n1 >> n2 >> n3;
   if (n1 < n2)
      if (n1 < n3) cout << "Their minimum is " << n1 << endl;
      else cout << "Their minimum is " << n3 << endl;
   else      // n1 >= n2
      if (n2 < n3) cout << "Their minimum is " << n2 << endl;
      else cout << "Their minimum is " << n3 << endl;
}
```

```
Enter two integers: 77 33 55
Their minimum is 33
```

در اجرای بالا، اولین شرط (n1<n2) نادرست است و سومین شرط (n2<n3) درست است. بنابراین گزارش می‌شود که n2 کمینه است.

این برنامه از برنامهٔ مثال 8–3 موثرتر است زیرا در هر اجرای آن فقط دو شرط سادهٔ تودرتو به جای سه شرط مرکب ارزیابی می‌شود ولی به نظر می‌رسد این برنامه ارزش کم‌تری نسبت به برنامهٔ مثال 8–3 داشته باشد زیرا منطق این برنامه پیچیده‌تر است. در مقایسه بین کارایی و سادگی، معمولا بهتر است سادگی انتخاب گردد.

× مثال 14–3 کمی پیچیده‌تر: یک بازی حدسی

برنامهٔ زیر عددی را که کاربر بین 1 تا 8 در ذهن دارد، پیدا می‌کند:

```
int main()
{  cout << "Pick a number from 1 to 8." << endl;
   char answer;
   cout << "Is it less than 5? (y|n): "; cin >> answer;
   if (answer = = 'y')            // 1 <=n <= 4
   {  cout << "Is it less than 3? (y|n): "; cin >> answer;
      if (answer = = 'y')         // 1 <= n <= 2
      {  cout << "Is it less than 2? (y|n): "; cin >> answer;
         if (answer = = 'y') cout << "Your number is 1."
             << endl;
         else cout << "Your number is 2." << endl;
      }
      else                        // 3 <= n <= 4
      {  cout << "Is it less than 4? (y|n): "; cin >> answer;
         if (answer = = 'y') cout << "Your number is 3."
             << endl;
         else cout << "Your number is 4." << endl;
      }
   }
   else                           // 5 <= n <= 8
   {  cout << "Is it less than 7? (y|n): "; cin >> answer;
      if (answer = = 'y')         // 5 <= n <= 6
```

```
      {  cout >> "Is it less than 6? (y|n): "; cin >> answer;
         if (answer = = 'y') cout << "Your number is 5."
             << endl;
         else cout << "Your number is 6." << endl;
      }
      else                          // 7 <= n <= 8
      {  cout << "Is it less than 8? (y n): "; cin >> answer;
         if (answer = = 'y') cout << "your number is 7."
             << endl;
         else cout << "Your number is 8." << endl;
      }
   }
}
```

برنامهٔ بالا با تجزیهٔ مسأله قادر است تنها با سه پرسش، هر یک از هشت عدد را پیدا کند. در اجرای زیر، کاربر عدد 6 را در نظر داشته است:

```
Pick a number from 1 to 8.
Is it less than 5? (y|n) : n
Is it less than 7? (y|n) : y
Is it less than 6? (y|n) : n
Your number is 6.
```

سعی کنید منطق برنامهٔ بالا را کشف کنید. به الگوریتم استفاده شده در مثال 14-3 الگوریتم ***جستجوی دودویی***[1] می‌گویند. این الگوریتم روی مجموعه‌های مرتب به کار می‌رود و به سرعت مشخص می‌کند آیا یک دادهٔ مفروض در این مجموعه هست یا خیر. در فصل‌های بعدی روش‌های دیگری از جستجو را خواهیم دید.

3-10 ساختار else if

دستور if..else تودرتو، اغلب برای بررسی مجموعه‌ای از حالت‌های متناوب یا موازی به کار می‌رود. در این حالات فقط عبارت else شامل دستور if بعدی خواهد بود. این قبیل کدها را معمولا با ساختار **else if** می‌سازند.

---

1 – Binary search

× مثال 15-3 استفاده از ساختار else if برای بررسی حالت‌های موازی

برنامهٔ زیر زبان کاربر را سوال می‌کند و سپس یک پیغام به همان زبان در خروجی چاپ می‌نماید:

```
int main()
{  char language;
   cout << "Engl., Fren., Ger., Ital., or Rus.? (e|f|g|i|r): ";
   cin >> language;
   if (language = = 'e') cout << "Welcome to ProjectC++.";
   else if (language = = 'f') cout << "Bon jour, ProjectC++.";
   else if (language = = 'g') cout << "Guten tag, ProjectC++.";
   else if (language = = 'i') cout << "Bon giorno, ProjectC++.";
   else if (language = = 'r') cout << "Dobre utre, ProjectC++.";
   else cout << "Sorry; we don't speak your language.";
}
```

```
Engl., Fren., Ger., Ital., or Rus.? (e|f|g|i|r): i
Bon giorno, ProjectC++.
```

این برنامه در حقیقت از دستور if..else تودرتو استفاده کرده. کد بالا را می‌توانستیم به ترتیب زیر بنویسیم:

```
if (language = = 'e') cout << "Welcome to ProjectC++.";
else
   if (language = = 'f') cout << "Bon jour, ProjectC++.";
   else
      if (language = = 'g') cout << "Guten tag, ProjectC++.";
      else
         if (language = = 'i') cout << "Bon giorno, ProjectC++.";
         else
            if (language = = 'r') cout << "Dobre utre, ProjectC++.";
            else cout << "Sorry; we don't speak your language.";
```

اما قالب قبلی به علت این که درک منطق برنامه را آسان‌تر می‌کند، بیشتر استفاده می‌شود. همچنین به تورفتگی کمتری احتیاج دارد.

× مثال 16-3 استفاده از ساختار else if برای مشخص کردن محدودهٔ نمره

برنامهٔ زیر یک نمرهٔ امتحان را به درجهٔ حرفی معادل تبدیل می‌کند:

```
int main()
{  int score;
   cout << "Enter your test score: ";  cin >> score;
   if (score > 100) cout << "Error: that score is out of range.";
   else if (score >= 90) cout << "Your grade is an A." << endl;
   else if (score >= 80) cout << "Your grade is a B." << endl;
   else if (score >= 70) cout << "Your grade is a C." << endl;
   else if (score >= 60) cout << "Your grade is a D." << endl;
   else if (score >= 0) cout << "Your grade is an F." << endl;
   else cout << "Error: that score is out of range.";
}
```

```
Enter your test score: 83
Your grade is a B.
```

مقدار متغیر score به شکل آبشاری با دستورهای انتخاب به طور متوالی بررسی می‌شود تا این که یکی از شرطها درست شود و یا به آخرین else برسیم.

3-11 دستورالعمل switch

دستور switch می‌تواند به جای ساختار else if برای بررسی مجموعه‌ای از حالت‌های متناوب و موازی به کار رود. نحو دستور switch به شکل زیر است:

```
switch (expression)
{  case constant1: statementlist1;
   case constant2: statementlist2;
   case constant3: statementlist3;
         :
         :
   case constantN: statementlistN;
   default: statementlist0;
}
```

این دستور ابتدا *expression* را برآورد می‌کند و سپس میان ثابت‌های **case** به دنبال مقدار آن می‌گردد. اگر مقدار مربوطه از میان ثابت‌های فهرست‌شده یافت شد، دستور *statementlist* مقابل آن **case** اجرا می‌شود. اگر مقدار مورد نظر میان **case**ها یافت نشد و عبارت **default** وجود داشت، دستور *statementlist* مقابل آن اجرا می‌شود. عبارت **default** یک عبارت اختیاری است. یعنی می‌توانیم در دستور **switch** آن را قید نکنیم. *expression* باید به شکل یک نوع صحیح ارزیابی شود و *constant*ها باید ثابت‌های صحیح باشند.

× مثال 17–3 نسخهٔ تغییریافته‌ای از مثال 16–3

این برنامه همان اثر برنامهٔ مثال 16–3 را دارد. در این نسخه از دستور switch استفاده شده:

```
int main()
{  int score;
   cout << "Enter your test score: ";  cin >> score;
   switch (score/10)
   {  case 10:
      case  9: cout << "Your grade is an A." << endl;  break;
      case  8: cout << "Your grade is a B." << endl;   break;
      case  7: cout << "Your grade is a C." << endl;   break;
      case  6: cout << "Your grade is a D." << endl;   break;
      case  5:
      case  4:
      case  3:
      case  2:
      case  1:
      case  0: cout << "Your grade is an F." << endl;  break;
      default: cout << "Error: score is out of range.\n";
   }
   cout << "Goodbye." << endl;
}
```

```
Enter your test score: 83
Your grade is a B.
```

```
Goodbye.
```

در برنامهٔ بالا ابتدا score بر 10 تقسیم می‌شود تا محدودهٔ اعداد بین صفر تا 10 محدود شود. بنابراین در اجرای آزمایشی، نمرهٔ 83 به 8 تبدیل می‌شود، اجرای برنامه به case 8 انشعاب می‌کند و خروجی مربوطه چاپ می‌گردد. سپس دستور break موجب می‌شود که اجرای برنامه از دستور switch خارج شده و به اولین دستور بعد از بلوک switch انشعاب کند. در آن‌جا عبارت "Goodbye." چاپ می‌شود.

لازم است در انتهای هر case دستور break قرار بگیرد. بدون این دستور، اجرای برنامه پس از این که case مربوطه را اجرا کرد از دستور switch خارج نمی‌شود، بلکه همهٔ caseهای زیرین را هم خط به خط می‌پیماید و دستورات مقابل آن‌ها را اجرا می‌کند. به این اتفاق، ***تلهٔ سقوط***[1] می‌گویند.

× مثال 18-3 تلهٔ سقوط در دستور switch

قصد بر این است که این برنامه مثل برنامهٔ 17-3 رفتار کند ولی بدون دستورهای break این برنامه دچار تلهٔ سقوط می‌شود:

```
int main()
{  int score;
   cout << "Enter your test score: ";  cin >> score;
   switch (score/10)
   {  case 10:
      case  9: cout << "Your grade is an A." << endl;  // LOGICAL ERROR
      case  8: cout << "Your grade is a B." << endl;   // LOGICAL ERROR
      case  7: cout << "Your grade is a C." << endl;   // LOGICAL ERROR
      case  6: cout << "Your grade is a D." << endl;   // LOGICAL ERROR
      case  5:
      case  4:
      case  3:
      case  2:
      case  1:
      case  0: cout << "Your grade is an F." << endl;  // LOGICAL ERROR
      default: cout << "Error: score is out of range.\n";
   }
```

1 – Fall-throw error

```
   cout << "Goodbye." << endl;
}
```

```
Enter your test score: 83
Your grade is a B.
Your grade is a C.
Your grade is a D.
Your grade is an F.
Error: score is out of range.
Goodbye.
```

در اجرای فوق پس از این که به case 8 انشعاب شد و عبارت مقابل آن چاپ شد، چون دستور break وجود ندارد، اجرای برنامه به خط بعدی یعنی case 7 می‌رود و عبارت "Your grade is a C." را نیز چاپ می‌کند و به همین ترتیب یکی یکی همهٔ عبارت‌های case را اجرا می‌نماید و سرانجام عبارت default را هم اجرا نموده و آنگاه از دستور switch خارج می‌شود.

3-12 عملگر عبارت شرطی

یکی از مزیت‌های C++ اختصار در کدنویسی است. ***عملگر عبارت شرطی*** یکی از امکاناتی است که جهت اختصار در کدنویسی تدارک دیده شده است. این عملگر را می‌توانیم به جای دستور if..else به کار ببریم. این عملگر از نشانه‌های **?** و **:** به شکل زیر استفاده می‌کند:

```
condition ? expression1 : expression2;
```

در این عملگر ابتدا شرط *condition* بررسی می‌شود. اگر این شرط درست بود، حاصل کل عبارت برابر با expression1 می‌شود و اگر شرط نادرست بود، حاصل کل عبارت برابر با expression2 می‌شود. مثلا در دستور انتساب زیر:

```
min = ( x<y ? x : y );
```

اگر x<y باشد مقدار x را درون min قرار می‌دهد و اگر x<y نباشد مقدار y را درون min قرار می‌دهد. یعنی به همین سادگی و اختصار، مقدار کمینهٔ x و y درون متغیر min قرار می‌گیرد. عملگر عبارت شرطی یک عملگر سه‌گانه است. یعنی سه عملوند را

برای تهیهٔ یک مقدار به کار می‌گیرد. بهتر است عملگر عبارت شرطی فقط در مواقع ضروری استفاده شود؛ وقتی که شرط و هر دو دستور خیلی ساده هستند.

× مثال 3-19 نسخهٔ جدیدی از برنامهٔ یافتن مقدار کمینه

این برنامه اثری شبیه برنامهٔ مثال 3-3 دارد:

```
int main()
{  int m, n;
   cout << "Enter two integers: ";
   cin >> m >> n;
   cout << ( m<n ? m : n ) << " is the minimum." << endl;
}
```

عبارت شرطی **(m<n ? m : n)** برابر با m می‌شود اگر m<n باشد و در غیر این صورت برابر با n می‌شود.

3-13 کلمات کلیدی[1]

اکنون با کلماتی مثل if و case و float آشنا شدیم. دانستیم که این کلمات برای C++ معانی خاصی دارند. از این کلمات نمی‌توان به عنوان نام یک متغیر یا هر منظور دیگری استفاده کرد و فقط باید برای انجام همان کار خاص استفاده شوند. مثلا کلمهٔ float فقط باید برای معرفی یک نوع اعشاری به کار رود. یک ***کلمهٔ کلیدی*** در یک زبان برنامه‌نویسی کلمه‌ای است که از قبل تعریف شده و برای هدف مشخصی منظور شده است. C++ استاندارد اکنون شامل 74 کلمهٔ کلیدی است:

and	**and_eq**	**asm**
auto	**bitand**	**Bitor**
bool	**break**	**case**
catch	**char**	**class**
compl	**const**	**const_cast**
continue	**default**	**delete**
dodouble	**dynamic_cast**	**else**
enum	**explicit**	**export**
extern	**dfalse**	**float**
for	**friend**	**goto**

1 – Keywords

```
if              inline              int
long            mutable             namespace
new             not                 not_eq
operator        or                  or_eq
privat          eprotected          public
register        reinterpret_cast    return
short           signed              sizeof
static          static_cast         struct
swich           template            this
throw           true                try
typedef         typoid              typename
using           union               unsigned
virtual         void                volatile
wchar_t         while               xor
xor_eq
```

کلمات کلیدی مانند if و else تقریبا در هر زبان برنامه‌نویسی پیدا می‌شوند. دیگر کلمات کلیدی همچون dynamic_cast منحصر به C++ هستند. 74 کلمهٔ کلیدی در C++ هست که همهٔ 32 کلمهٔ کلیدی زبان C را نیز شامل می‌شود.

دو نوع کلمهٔ کلیدی وجود دارد: ***کلمه‌های رزرو شده*** و ***شناسه‌های استاندارد***. یک کلمهٔ رزرو شده کلمه‌ای است که یک دستور خاص از آن زبان را نشان می‌دهد. کلمهٔ کلیدی if و else کلمات رزرو شده هستند. یک شناسهٔ استاندارد کلمه‌ای است که یک نوع دادهٔ استاندارد از زبان را مشخص می‌کند. کلمات کلیدی bool و int شناسه‌های استاندارد هستند زیرا هر یک از آن‌ها یک نوع دادهٔ خاص را در زبان C++ مشخص می‌کنند. برای اطلاعات بیشتر در مورد کلمات کلیدی C++ به مراجع این زبان یا راهنمای کامپایلرتان مراجعه کنید.

پرسش‌های گزینه‌ای

1 – به ازای کد `if (k = 0) i=k;` کدام جمله صحیح است؟

الف) اگر k مساوی با صفر باشد، آنگاه مقدار k در i کپی می‌شود.

ب) اگر k مساوی با غیر صفر باشد، آنگاه مقدار k در i کپی می‌شود.

ج) کامپایلر خطا می‌گیرد زیرا عملگر برابری == است نه = .

د) به ازای همهٔ مقادیر k مقدار k در i کپی می‌شود.

2 – اگر متغیر b از نوع بولین باشد، کد `if(b) b;` چه کاری انجام می‌دهد؟

الف) اگر b برابر با true باشد، آنگاه b را false می‌کند.

ب) اگر b برابر با false باشد، آنگاه b را true می‌کند.

ج) اگر b برابر با true باشد، آنگاه b را false می‌کند وگرنه b را true می‌کند.

د) اگر b برابر با false باشد، آنگاه b را true می‌کند وگرنه b را false می‌کند.

3 – کد `if (i==0) if (j==0) k=0;` معادل کدام یک از کدهای زیر است؟

الف)
```
if ((i==0) || (j==0)) k=0;
```

ب)
```
if ((i==0) && (j==0)) k=0;
```

ج)
```
if (i==0) k=0;
```

د)
```
if (j==0) k=0;
```

4 – در ارزیابی عبارت‌های شرطی:

الف) صفر به معنای درست و هر مقدار غیر صفر به معنای نادرست است.

ب) صفر به معنای نادرست و هر مقدار غیر صفر به معنای درست است.

ج) یک به معنای درست و هر مقدار غیر یک به معنای نادرست است.

د) یک به معنای نادرست و هر مقدار غیر یک به معنای درست است.

5 – اگر m یک متغیر بولین باشد، در کد `if(m) i++ else i--;` چه روی می‌دهد؟

الف) اگر m برابر با true باشد، به i یک واحد افزوده می‌شود وگرنه از i یک واحد کاسته می‌شود

ب) اگر m برابر با false باشد، به i یک واحد افزوده می‌شود وگرنه از i یک واحد کاسته می‌شود

ج) اگر m برابر با true باشد، به i دو واحد افزوده می‌شود وگرنه از i دو واحد کاسته می‌شود

د) اگر m برابر با False باشد، به i دو واحد افزوده می‌شود وگرنه از i دو واحد کاسته می‌شود

6 – در کد `if (i<0) i++; j++;` چه رخ می‌دهد؟

الف) اگر i از صفر کوچک‌تر باشد، به j یک واحد افزوده می‌شود

ب) اگر i بزرگ‌تر یا مساوی صفر باشد، به j یک واحد افزوده می‌شود

ج) اگر به i یک واحد افزوده شود، به j هم یک واحد افزوده می‌شود.

د) مقدار i ربطی به j ندارد و در هر حال به j یک واحد افزوده می‌شود.

7 – در کد `if (i<j) {i++; j--;}` چه اتفاقی می‌افتد؟

الف) اگر i از j کوچک‌تر باشد، از j یک واحد کاسته می‌شود

ب) اگر i از j بزرگ‌تر باشد، از j یک واحد کاسته می‌شود

ج) در هر حال به i یک واحد اضافه می‌شود و ربطی به j ندارد

د) در هر حال از j یک واحد کاسته می‌شود و ربطی به i ندارد

8 – خروجی کد مقابل چیست؟

```
int n=55;
{  int n=77;
   cout << n << endl;
}
cout << n;
```

الف) روی اولین سطر 55 چاپ می‌شود و روی دومین سطر 77 چاپ می‌شود

ب) روی اولین سطر 77 چاپ می‌شود و روی دومین سطر 55 چاپ می‌شود

ج) روی هر دو سطر مقدار 55 چاپ می‌شود

د) روی هر دو سطر مقدار 77 چاپ می‌شود

9 – حاصل اجرای کد `if ((d>1) || (d/m)) d++;` به ازای d=2 و m=0 چیست؟

الف) خطای تقسیم بر صفر رخ می‌دهد و برنامه متوقف می‌شود

ب) شرط دستور if نادرست است پس دستور d++ نادیده گرفته می‌شود

ج) به d یک واحد اضافه می‌شود

د) خطای تقسیم بر صفر رخ می‌دهد پس دستور d++ نادیده گرفته می‌شود

10 – اگر a=0 و b=1 و c=2 باشد، مقدار c پس از اجرای کد زیر، چیست؟

```
if (a==1)
if (b==1) c++;
else c--;
```

الف) c = 3 ب) c = 1 ج) c = 2 د) c = 4

11 – اگر i = 5 باشد، مقدار i پس از اجرای کد زیر، چیست؟

```
switch (i)
{  case 5: i++;
   case 0: i--;
   default: i--;
}
```

الف) i = 6 ب) i = 5 ج) i = 4 د) i = 3

12 – تله سقوط وقتی رخ می‌دهد که :

الف) به جای عملگر برابری (==) از عملگر جایگزینی (=) استفاده کنیم

ب) در دستور if پرانتزهای شرط را فراموش کنیم

ج) تلاش کنیم عددی را بر صفر تقسیم کنیم

د) در دستور switch دستورهای break را فراموش کنیم

13 – کد if (x>y) p=x; else p=y; معادل کدام یک از کدهای زیر است؟

الف) p=(x>y ? y : x); ب) p=(x>y ? x : y);

ج) (x>y ? p=y : p=x); د) (x>y ? p=x : p=y);

14 – معنی جملهٔ مقابل چیست؟ «عملگر عبارت شرطی یک عملگر سه گانه است»

الف) یعنی عملگر عبارت شرطی سه شکل متفاوت دارد

ب) یعنی عملگر عبارت شرطی سه کاربرد متفاوت دارد

ج) یعنی عملگر عبارت شرطی ترکیبی از سه شرط است

د) یعنی عملگر عبارت شرطی سه عملوند را برای تهیهٔ یک مقدار به کار می‌گیرد

15 – کدام دستور زیر، یک دستور انتخاب نیست؟

الف) دستور break ب) دستور if

ج) دستور if..else د) دستور switch

پرسش‌های تشریحی

1- یک دستورالعمل منفرد در C++ بنویسید که اگر متغیر cout از 100 تجاوز کرد عبارت "Too many" را چاپ کند.

2- چه اشتباهی در کدهای زیر است؟

a.
```
cin << count;
```
b.
```
if x < y min = x
   else min = y;
```

3- چه اشتباهی در این کد برنامه وجود دارد؟

```
cout << "Enter n: ";
cin >> n;
if (n < 0)
   cout << "That is negative. Try again." << endl;
   cin >> n;
else
   cout << "o.k. n = " << n << endl;
```

4- چه تفاوتی بین کلمهٔ رزرو شده و شناسهٔ استاندارد است؟

5- مشخص کنید هر یک از عبارات زیر درست است یا نادرست. اگر نادرست است بگویید چرا؟

الف – عبارت (p||q) ! با عبارت !q || !p برابر است.

ب – عبارت p!!! با عبارت !p برابر است.

ج – عبارت p && q || r با عبارت p && (q || r) برابر است.

6- برای هر یک از عبارت‌های بولی زیر یک جدول درستی بسازید که مقادیر درستی آن‌ها را (0 یا 1) به ازای هر مقدار از عملوندهای p و q نشان دهد:

الف – !p || q

ب – !p&&!q || p&&q

ج – !(p&&q) && (p || q)

7- با استفاده از جدول درستی تعیین کنید که آیا دو عبارت بولی در هر یک از معادلات زیر برابرند یا خیر؟

الف – !q && !p و !(p && q)

ب – p و !!p

ج – !q || p و !p || q

د – r && (p && q) و p && (q && r)

ه – r && (p || q) و p || (q && r)

8- ارزیابی میانبری چیست و چه فایده‌ای دارد؟

9- چه اشتباهی در کد زیر است؟

```
if (x = 0) cout << x << " = 0\n";
else cout << x << " != 0\n";
```

10- چه اشتباهی در کد زیر وجود دارد؟

```
if (x < y < z) cout << x << " < " << y << " < " <<
z << endl;
```

11- برای هر یک از شرط‌های زیر یک عبارت منطقی بسازید:

الف – score بزرگ‌تر یا مساوی 80 و کوچک‌تر از 90 باشد

ب – answer برابر با 'n' یا 'N' باشد

ج – n یک عدد زوج باشد ولی برابر با 8 نباشد

د – ch یک حرف بزرگ (capital) باشد

12- برای هر یک از شرط‌های زیر یک عبارت منطقی بسازید:

الف – n بین 0 و 7 باشد ولی برابر با 3 نباشد

ب – n بین 0 و 7 باشد ولی زوج نباشد

ج – n بر 3 بخش‌پذیر باشد ولی بر 30 بخش‌پذیر نباشد

د – ch یک حرف بزرگ یا کوچک باشد

13- چه اشتباهی در این کد است؟

```
if (x = = 0)
  if (y = = 0) cout << "x and y are both zero."
                    << endl;
else cout << "x is not zero." << endl;
```

14- چه تفاوتی بین دو دستورالعمل زیر است؟

a.
```
if (n > 2) { if (n < 6) cout << "OK"; }
  else cout << "NG";
```

b.
```
if (n > 2) { if (n < 6) cout << "OK" ;
                else cout << "NG"; }
```

15- تلهٔ سقوط چیست؟

16- عبارت زیر چگونه ارزیابی می‌شود؟

```
(x < y ? -1 : (x == y ? 0 : 1) );
```

17- یک دستورالعمل منفرد در C++ بنویسید که با استفاده از عملگر عبارت شرطی، قدرمطلق x را در متغیر absx قرار دهد.

18- یک دستورالعمل منفرد در C++ بنویسید که اگر متغیر count از 100 تجاوز کرد عبارت "Too many" را چاپ کند با استفاده از:

الف - یک دستورالعمل if

ب - یک عملگر عبارت شرطی

تمرین‌های برنامه‌نویسی

1- برنامهٔ مثال 1-3 را طوری تغییر دهید که تنها اگر n بر d قابل تقسیم باشد پاسخی را چاپ کند.

2- برنامهٔ مثال 5-3 را طوری تغییر دهید که کمینهٔ چهار عدد صحیح را چاپ کند.

3- برنامهٔ مثال 5-3 را طوری تغییر دهید که حد وسط سه عدد صحیح وارد شده را چاپ کند.

4- برنامهٔ مثال 6-3 را طوری تغییر دهید که همان اثر را داشته باشد اما بدون استفاده از بلوک دستورالعمل.

5- پیش‌بینی کنید خروجی برنامهٔ مثال 17-3 چیست اگر اعلان خط پنجم برنامه را پاک کنیم. برنامهٔ تغییریافته را برای بررسی پیش‌بینی خود اجرا کنید.

6- برنامه‌ای نوشته و اجرا کنید که سن کاربر را بخواند و اگر سن کوچک‌تر از 18 بود عبارت "You are a child" را چاپ کند و اگر سن بین 18 و 65 بود عبارت "You are an adult" را چاپ کند و اگر سن بزرگ‌تر یا مساوی 65 بود عبارت "you are a cenior citizen" را چاپ کند.

7- برنامه‌ای نوشته و اجرا کنید که دو عدد صحیح را می‌خواند و با استفاده از یک عملگر عبارت شرطی، با توجه به این که آیا یکی از این دو مضرب دیگری است یا خیر، عبارت "multiple" یا "not" را چاپ کند.

8- برنامه‌ای نوشته و اجرا کنید که یک ماشین‌حساب ساده را شبیه‌سازی می‌کند که دو عدد صحیح و یک کاراکتر را می‌خواند و سپس اگر کاراکتر (+) باشد مجموع را چاپ کند و اگر کاراکتر (-) باشد تفاضل را چاپ کند و اگر کاراکتر(*) باشد حاصل‌ضرب را چاپ کند و اگر کاراکتر (/) باشد حاصل‌تقسیم را چاپ کند و اگر کاراکتر (%) باشد باقیمانده تقسیم را چاپ کند. از یک دستورالعمل switch استفاده کنید.

9- برنامه‌ای نوشته و اجرا کنید که بازی "سنگ – کاغذ – قیچی" را انجام دهد. در این بازی دو نفر به طور هم‌زمان یکی از عبارات "سنگ" یا "کاغذ" یا "قیچی" را می‌گویند (و یا یکی از علامت‌های از قبل مشخص را با دست نشان می‌دهند). برنده کسی است که شیء غلبه‌کننده بر دیگری را انتخاب کرده باشد. حالات ممکن، چنین است که کاغذ

بر سنگ غلبه می‌کند (می‌پوشاند)، سنگ بر قیچی غلبه می‌کند (می‌شکند) و قیچی بر کاغذ غلبه می‌کند (می‌برد). برای اشیاء از یک نوع شمارشی استفاده کنید.

10- مسألهٔ 9 را با استفاده از دستور switch حل کنید.

11- مسألهٔ 9 را با استفاده از عبارات شرطی حل کنید.

12- برنامه‌ای را نوشته و اجرا کنید که یک معادلهٔ درجه دوم را حل می‌کند. معادلهٔ درجه دوم معادله‌ای است که به شکل $ax^2+bx+c=0$ باشد. a و b و c ضرایب هستند و x مجهول است. ضرایب، اعداد حقیقی هستند که توسط کاربر وارد می‌شوند. بنابراین باید از نوع float یا double اعلان گردند. از آن‌جا که معادلهٔ درجه دوم معمولا دو ریشه دارد، برای جواب‌ها از x1 و x2 استفاده کنید. جواب‌ها باید از نوع double اعلان گردند تا از خطای گرد کردن جلوگیری شود.

13- برنامه‌ای را نوشته و اجرا کنید که یک عدد شش رقمی را می‌خواند و مجموع شش رقم آن عدد را چاپ می‌کند. از عملگر تقسیم (/) و عملگر باقیمانده (٪) برای بیرون کشیدن رقم‌ها از عدد ورودی استفاده کنید. برای مثال اگر عدد ورودی n برابر با 876,543 باشد، آنگاه n/1000%10 برابر با رقم یکان هزار یعنی 6 است.