فصل ششم – ساختارهای کنترلی

ساختارهای تصمیم

- دستور if
- دستور if…else
- عملگر سه تایی
- دستور if چندگانه
- دستور if تو در تو
- عملگرهای منطقی
- دستور switch

تکرار

- while
- do while
- for
- حلقه‌های تو در تو (Nested Loops)
- خارج شدن از حلقه با استفاده از break و continue

دستور if

```
if ( شرط مورد نظر )
        دستور مورد نظر ;
```

به مثال زیر توجه کنید:

```
if (x == 50)
     cout << "x is 50";
```

```
if ( x==50 )
  {
    cout << "x is ";
    cout << x;
  }
```

```
if ( x == 50)
     cout << "x is ";
     cout << x ;
```

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.

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
```

مثال :

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6         //Declare a variable and set it a value less than 10
7         int number = 5;
8
9         //If the value of number is less than 10
10        if (number < 10)
11              cout << "Hello World." << endl;
12
13        //Change the value of a number to a value which is greater than 10
14        number = 15;
15
16        //If the value of number is greater than 10
17        if (number > 10)
18              cout << "Goodbye World.";
19  }
```

```
Hello World.
Goodbye World.
```

دستور if…else

```
if (شرط مورد نظر)
        دستور۱ ;
else
        دستور۲ ;
```

بیشتر از یک دستور :

```
if ( x > 50 )
   {
     cout << x;
     cout << "is greater than 50";
   }
else
   {
     cout << x;
     cout << "is less than 50";
   }
```

مثال : این برنامه بررسی می کند که یک عدد صحیح مثبت بر عدد دیگر قابل تقسیم است یا خیر:

```
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
```

تمرین : برنامه ای بنویسید که مشخص کند 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
```

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
```

یافتن کمینه ۳ عدد:

```
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
```

دستور if چندگانه

اگر بخواهید چند شرط را بررسی کنید چکار می‌کنید؟ می‌توانید از چندین دستور if استفاده کنید و بهتر است که این دستورات if را به صورت زیر بنویسید:

```
if (condition)
{
   code to execute;
}
else if (condition)
{
   code to execute;
}
else if (condition)
{
   code to execute;
}

else
{
   code to execute;
}
```

مثال :

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

int main()
{
    int choice;

    cout << "What's your favorite color?" << endl;
    cout << "[1] Black" << endl;
    cout << "[2] White" << endl;
    cout << "[3] Blue" << endl;
    cout << "[4] Red" << endl;
    cout << "[5] Yellown" << endl;

    cout << "Enter your choice: ";
    cin >> choice;

    if (choice == 1)
    {
        cout << "You might like my black t-shirt." << endl;
    }
    else if (choice == 2)
    {
        cout << "You might be a clean and tidy person." << endl;
    }
    else if (choice == 3)
    {
        cout << "You might be sad today." << endl;
    }
    else if (choice == 4)
    {
        cout << "You might be inlove right now." << endl;
    }
    else if (choice == 5)
    {
        cout << "Lemon might be your favorite fruit." << endl;
    }
    else
    {
        cout << "Sorry, your favorite color is not in the choices above." << endl;
    }

}
```

```
What's your favorite color?
[1] Black
[2] White
[3] Blue
[4] Red
[5] Yellow

Enter your choice: 1
You might like my black t-shirt.
What's your favorite color?
[1] Black
[2] White
[3] Blue
[4] Red
[5] Yellow

Enter your choice: 999
Sorry, your favorite color is not in the choices above.
```

دستور if تو در تو

می‌توان از دستور if تو در تو در C++ استفاده کرد. یک دستور ساده if در داخل دستور if دیگر.

```
#include <iostream.h>
int main( )
{
  int x;
  cout << "Please enter a number:";
  cin >> x;

  if ( x > 0 )
     cout << x << "is positive.";
  else
     if ( x < 0 )
        cout << x << "is negative.";
     else
        cout << "The number that you entered is 0.";
  return 0;
}
```

برنامه فوق را سه بار با سه عدد مختلف اجرا می کنیم. خروجی ها به صورت زیر می باشند:

```
Please enter a number : 10
10 is positive.
Please enter a number : -5
-5 is negative.
Please enter a number : 0
The number that you entered is 0.
```

نکته : برای وضوح برنامه پیشنهاد می شود همانند برنامه فوق هنگام استفاده از **if** یا **if/else** و یا دیگر ساختارهای کنترلی از تورفتگی های مناسب استفاده کنید. یعنی به عنوان مثال دستور **if** را به صورت زیر:

```
if ( x > 0 )
    cout << x << "is positive.";
```

بنویسیم و نه به صورت زیر :

```
if ( x > 0 )
cout << x << "is positive.";
```

استفاده از عملگرهای منطقی

عملگرهای منطقی به شما اجازه می‌دهند که چندین شرط را با هم ترکیب کنید. این عملگرها حداقل دو شرط را در گیر می‌کنند و در آخر یک مقدار بولی را بر می‌گردانند. در جدول زیر برخی از عملگرهای منطقی آمده است :

عملگر	تلفظ	مثال	تأثیر
&&	And	z = (x > 2) && (y < 10)	مقدار z درصورتی true است که هر دو شرط دو طرف عملگر مقدارشان true باشد .اگر فقط مقدار یکی از شروط false باشد مقدار z ، false خواهد شد .
||	Or	z = (x > 2) || (y < 10)	مقدار z درصورتی true است که یکی از دو شرط دو طرف عملگر مقدارشان true باشد. اگر هر دو شرط مقدارشان false باشد مقدار z ، false خواهد شد .
!	Not	z = !(x > 2)	مقدار Z در صورتی true است که مقدار شرط falseباشد و در صورتی falseاست که مقدار شرط true باشد .

یافتن کمینه ۳ عدد:

```
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
```

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
```

دستور Switch

در C++ ساختاری به نام switch وجود دارد که به شما اجازه می‌دهد که با توجه به مقدار ثابت یک متغیر چندین انتخاب داشته باشید. دستور switchمعادل دستور if تو در تو است با این تفاوت که در دستور switch متغیر فقط مقادیر ثابتی از اعداد ، رشته‌ها و یا کاراکترها را قبول می‌کند . مقادیر ثابت مقادیری هستند که قابل تغیر نیستند. در زیر نحوه استفاده از دستور switch آمده است:

```
switch (عبارتی که باید مورد بررسی قرار گیرد)
   {
      case مقدار ثابت ۱ :
                  مجموعه دستورات ۱
                  break;
      case مقدار ثابت ۲ :
                  مجموعه دستورات ۲
                  break;

      .
      .
      .

      case  n  مقدار ثابت :
                  n مجموعه دستورات
                  break;
      default :
          مجموعه دستورات حالت پیش فرض
 }
```

مثال :

```
#include <iostream.h>
int main( )
{
  int x;
  cout << "Please enter a number:";
  cin >> x;

  switch (x) {
    case 1:
      cout << "x is 1";
      break;
    case 2:
      cout << "x is 2";
      break;
    default:
      cout << "Unknown value";
  }
  return 0;
}
```

برنامه فوق را سه بار با سه عدد مختلف اجرا می کنیم. خروجی ها به صورت زیر می باشند:

```
Please enter a number:1
x is 1
Please enter a number:2

x is 2
Please enter a number:5
Unknown value
```

مثال :

```
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.
```

تمرین :

برنامه ای بنویسید که ۲ عدد **a,b** ویک عملگر محاسباتی / *- + را از ورودی خوانده و عمل مناسب را با توجه به عملگر وارد شده انجام دهد ؟

تمرین :

برنامه ای بنویسید که ۲ عدد **a,b** ویک عملگر محاسباتی / * - + را از ورودی خوانده و عمل مناسب را با توجه به عملگر وارد شده انجام دهد ؟

```
#include <iostream.h>
#include <conio.h>
void  main (void)
{
        int a,b;
        char c;
        cin >> a >> b >> c;
        switch (c)
        {
        case '+' :
                cout << "a+b =" << a+b;
                break;
        case '-' :
                cout << "a-b =" << a-b;
                break;
        case '*' :
                cout << "a*b =" << a*b;
                break;
        case '/' :
                cout << "a/b =" << a/b;
                break;
        }
        getch( );
}//end case
```

حلقه While

ابتدایی‌ترین ساختار تکرار در C++ حلقه While است.ابتدا یک شرط را مورد بررسی قرار می‌دهد و تا زمانیکه شرط برقرار باشد کدهای درون بلوک اجرا می‌شوند. ساختار حلقه Whileبه صورت زیر است:

```
while ( شرط مورد نظر )
        {
            مجموعه دستورات
        }
```

در برنامه زیر نحوه استفاده از حلقه while آمده است:

```
#include <iostream>

using namespace std;

int main()
{
   int counter = 1;

    while (counter <= 10)
    {
       cout << "Hello World!" << endl;
       counter++;
    }
}
```

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

اگر بخواهید یک حلقه بی نهایت ایجاد کنید که هیچ گاه متوقف نشود باید یک شرط ایجاد کنید که همواره درست true باشد.

```
while(true)
{
   //code to loop
}
```

این تکنیک در برخی موارد کارایی دارد و آن زمانی است که شما بخواهید با استفاده از دستورات break و return که در آینده توضیح خواهیم داد از حلقه خارج شوید.

می خواهیم برنامه ای بنویسیم که اولین توانی از عدد ۲ که بزرگتر از ۱۰۰۰ باشد را در خروجی چاپ کند:

```
#include <iostream.h>
int main( )
 {
  int product = 2;
  while (product <= 1000)
       product = 2 * product;

  cout << "The first power of 2 larger than 1000 is "
       <<product <<endl;
  return 0;
 }
```

برنامه ای بنویسید که مجموع اعداد یک تا صد را محاسبه و در خروجی چاپ کند:

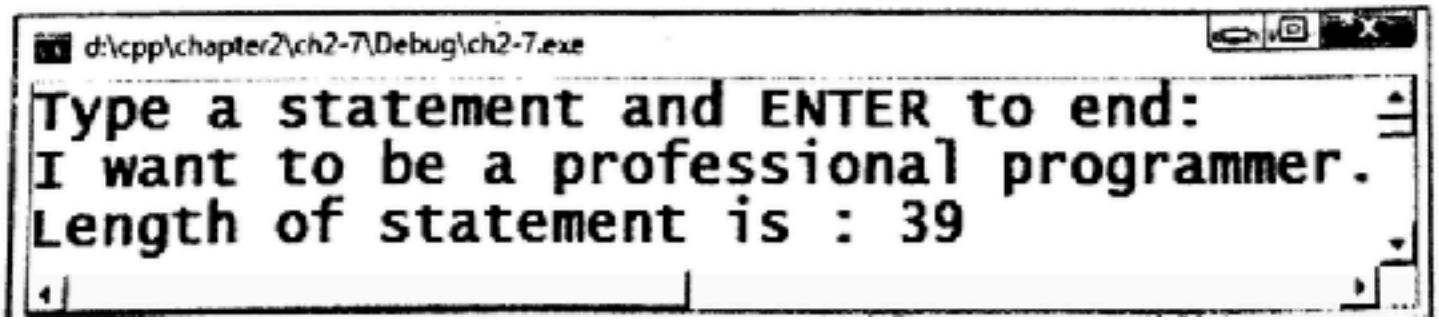
```
#include <iostream.h>
int main( )
  {
    int n=1, sum=0;
    while (n <= 100)
      {
        sum += n;          // sum = sum + n;
        ++n;               // n = n + 1;
      }
    cout << "1 + 2 + ... + 100 =" <<sum << endl;
    return 0;
  }
```

مثال ۷-۲ برنامه‌ای که جمله‌ای را از ورودی خوانده، تعداد کاراکترهای جمله را شمارش می‌کند. انتهای جمله به کلید Enter ختم می‌شود. در این برنامه، count تعداد کاراکترهای خوانده‌شده از ورودی است.

۱. برنامه‌ی زیر را در فایل 2-7.cpp تایپ کنید:

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main()
{
    int count = 0 ;
    cout << "Type a statement and ENTER to end:" << endl;
    while(cin.get() != '\n')
       count ++ ;
    cout << "Length of statement is : " << count ;
    cin.get(); // ignore end of line char
    cin.get();
}
```

۲. برنامه را اجرا کنید تا خروجی آن را به صورت زیر ببینید:



حلقه do while

حلقه do while یکی دیگر از ساختارهای تکرار است. این حلقه بسیار شبیه حلقه while است با این تفاوت که در این حلقه ابتدا کد اجرا می شودو سپس شرط مورد بررسی قرار می‌گیرد. ساختار حلقه do while به صورت زیر است:

```
do {
      مجموعه دستورات
}while ( شرط مورد نظر );
```

مثال :

```
int number = 1;
do
{
    cout << "Hello World!" << endl;
} while (number > 10);
```

```
Hello World!
```

با اجرای کد بالا، اول دستورات بلوک do اجرا می‌شوند و بعد مقدار number با عدد ۱۰ مقایسه می‌شود. در نتیجه حتی اگر شرط نادرست باشد باز هم قسمت doحداقل یک بار اجرا می‌شوند.

```
int number = 1;
while (number > 10)
{
    cout << "Hello World!" << endl;
}
```

اما در کد بالا چون اول مقدار number ابتدا مورد مقایسه قرار می‌گیرد، اگر شرط درست نباشد دیگر کدی اجرا نمی‌شود. یکی از موارد برتری استفاده از حلقه do while نسبت به حلقه while زمانی است که شما بخواهید اطلاعاتی از کاربر دریافت کنید.

در دو کد زیر، یک عملیات یکسان توسط دو حلقه while و do while پیاده سازی شده است:

```
//while version

cout << "Enter a number greater than 10: " << endl;
cin >> number;

while (number < 10)
{
    cout << "Enter a number greater than 10: " << endl;
    cin >> number;
}
```

```
//do while version

do
{
    cout << "Enter a number greater than 10: " << endl;
    cin >> number;
} while (number < 10);
```

مشاهده می‌کنید که از کدهای کمتری در بدنه do while نسبت به while استفاده شده است.

به مثال زیر توجه نمایید و خروجی آن را معلوم کنید :

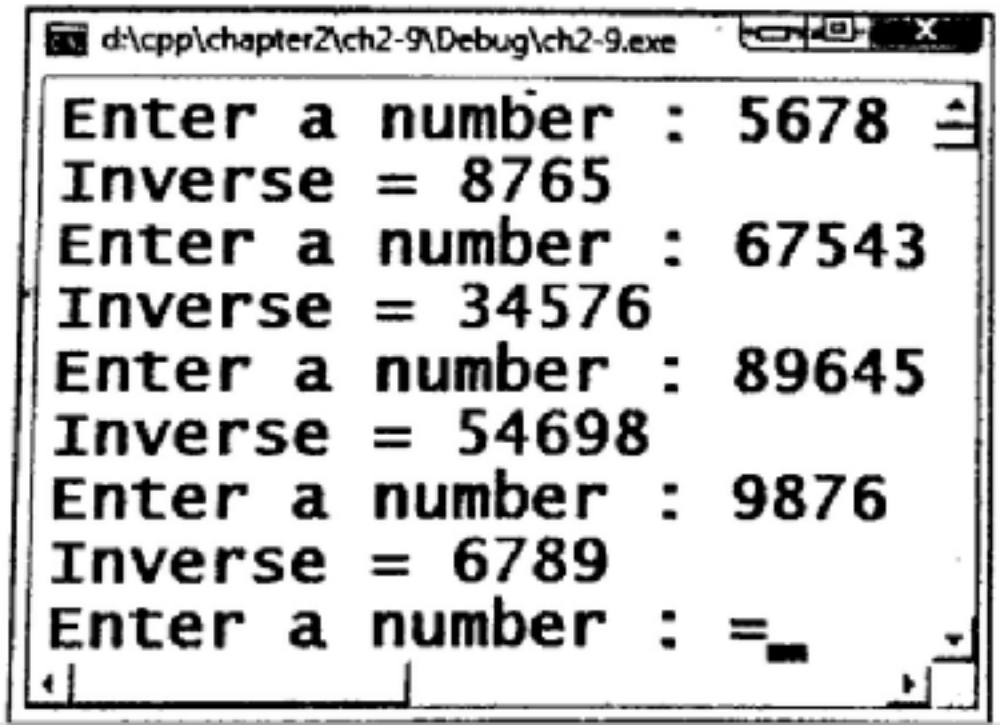
```
#include <iostream.h>

int main()
{
    int counter = 1;
    do {
        cout << counter << " ";
       }while ( ++counter <= 10 );
    cout << endl;

    return 0;
}
```

مثال ۹-۲ برنامه‌ای که تعدادی عدد را از ورودی خوانده، وارون انها را به خروجی می‌برد. وارون عددی مثل x ، عددی مثل y است به طوری که ارقام عدد x از راست به چپ مثل ارقام عدد y از چپ به راست باشد. به عنوان مثال ، وارون عدد ۲۱۵۳ ، عدد ۳۵۱۲ است.

۱. برنامه‌ی زیر را در فایل 2-9.cpp تایپ کنید:

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main()
{
    int num, digit ;
    while(true) {
       cout << "\n Enter a number : ";
       cin >> num;
       cout << " Inverse = ";
       do {
          digit = num % 10;
          cout << digit;
          num /= 10;
       } while(num != 0);
    }//end of while(true)
}
```

۲. برنامه را اجرا کنید تا خروجی آن را به صورت زیر ببینید:



حلقه for

یکی دیگر از ساختارهای تکرار حلقه for است. این حلقه عملی شبیه به حلقه while انجام می‌دهد و فقط دارای چند خصوصیت اضافی است. ساختار حلقه for به صورت زیر است:

```
for(initialization; condition; operation)
{
   code to repeat;
}
```

مقدار دهی اولیه (initialization) اولین مقداری است که به شمارنده حلقه می‌دهیم.شمارنده فقط در داخل حلقه for قابل دسترسی است.

شرط (condition) در اینجا مقدار شمارنده را با یک مقدار دیگر مقایسه می‌کند و تعیین می‌کند که حلقه ادامه یابد یا نه.

عملگر (operation) که مقدار اولیه متغیر را کاهش یا افزایش می‌دهد.

در زیر یک مثال از حلقه for آمده است:

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

int main()
{
    for (int i = 1; i <= 10; i++)
    {
        cout << "Number " << i << endl;
    }
}
```

```
Number 1
Number 2
Number 3
Number 4
Number 5
Number 6
Number 7
Number 8
Number 9
Number 10
```

اگر بخواهیم اعداد معکوس در خروجی چاپ شوند :

```
for (int i = 10; i > 0; i--)
{
    //code omitted
}
```

می توان قسمت شرط و عملگر را به صورت های دیگر تغییر داد :

```
for (int i = 1, y = 2; i < 10 && y > 20; i++, y -= 2)
{
    //some code here

}
```

به این نکته توجه کنید که اگر از چندین متغیر شمارنده یا عملگر در حلقه for استفاده می‌کنید باید آن‌ها را با استفاده از کاما از هم جدا کنید.

ساختار تکرار **for** را با ساختار تکرار **while** نیز می توان پیاده سازی کرد به عنوان مثال دو برنامه زیر اعداد ۱ تا ۵ را بر روی صفحه نمایش چاپ می کنند:

```
#include <iostream>

int main()
{
  int counter = 1;

  while ( counter <= 5 ) {
     cout << counter << endl;
     ++counter;
  }
  return 0;
}
```

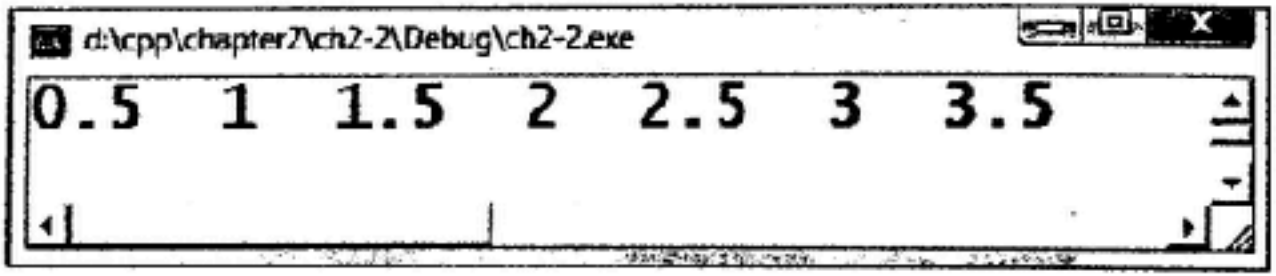
```
#include <iostream>

int main()
{

  for ( int counter = 1; counter <= 5; counter++ )
     cout << counter << endl;

return 0;
}
```

مثال ۲-۲ برنامه‌ای که اعداد 0.5 تا 3.5 را با فاصله‌ی 0.5 در خروجی چاپ می‌کند. هدف از این برنامه، آشنایی با حلقه‌ی تکرار با اندیس اعشاری است. چون فقط یک دستور در حلقه تکرار می‌شود، نیاز به } و { نیست.

۱. برنامه‌ی زیر را در فایل 2-2.cpp تایپ کنید:

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main()
{
      float i ;
      for(i = 0.5; i <= 3.5; i += 0.5)
            cout <<  i << "  ";
      cin.get();
}
```

۲. برنامه را اجرا کنید تا خروجی آن را به صورت زیر ببینید:



مثال ۳-۲ برنامه‌ای که کاراکترهای "a" تا "f" را به همراه کد اسکی آن‌ها به خروجی می‌برد.

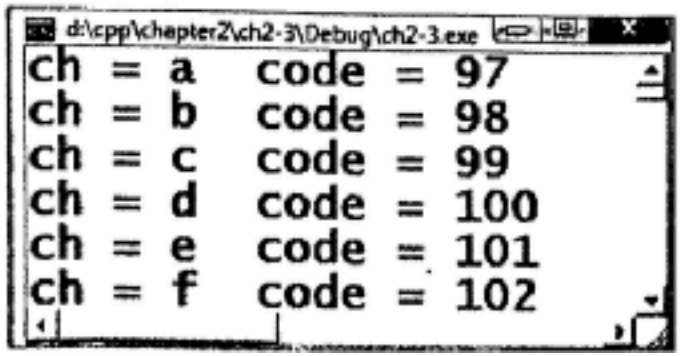
شرح مثال ۳-۲

در این برنامه، از یک حلقه‌ی تکرار استفاده شد که اندیس آن کاراکتری است. برای نوشتن کد اسکی کاراکتر، آن را به یک متغیر صحیح نسبت می‌دهیم و آن متغیر را چاپ می‌کنیم.

۱. برنامه‌ی زیر را در فایل 2-3.cpp تایپ کنید:

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main()
{
    char ch;
    int code;
    for(ch = 'a'; ch <= 'f'; ch ++) {
        code = ch;
        cout  << "ch = " << ch << "  code = " << code << endl;
    }
    cin.get();
}
```

۲. برنامه را اجرا کنید تا خروجی آن را به صورت زیر ببینید:

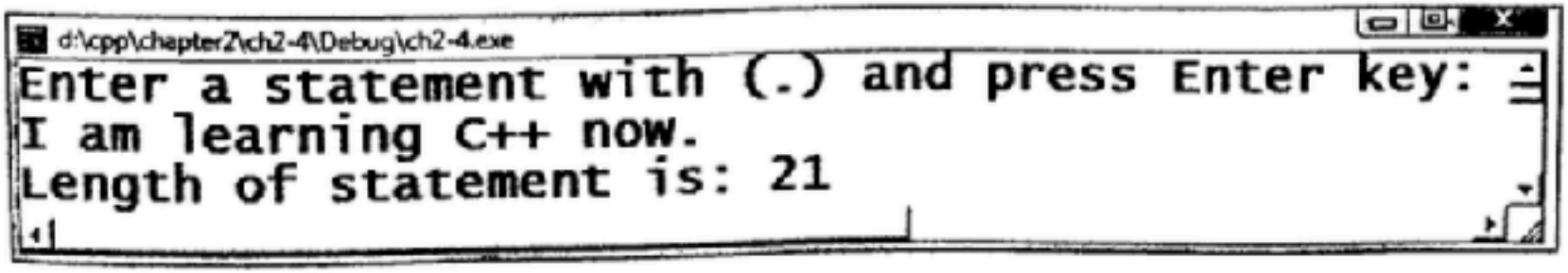


مثال ۴-۲ برنامه‌ای که جمله‌ای را از ورودی می‌خواند و تعداد حروف آن را شمارش می‌کند. انتهای جمله به نقطه ختم می‌شود. در این برنامه، ch ، کاراکتری است که از ورودی خوانده می‌شود و count تعداد کاراکترهای خوانده‌شده است. پس از واردکردن نقطه به عنوان آخرین کاراکتر، کلید Enter را فشار دهید.

۱. برنامه‌ی زیر را در فایل 2-4.cpp تایپ کنید:

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main()
{
    int count;
    cout << "Enter a statement with (.) and press Enter key:"
         << endl ;
    for(count = 0; cin.get() != '.' ; count++);
    cout << "Length of statement is: " << count ;
    cin.get(); // ignore end of line char
    cin.get();
}
```

۲. برنامه را اجرا کنید تا خروجی آن را به صورت زیر ببینید:



حلقه‌های تو در تو (Nested Loops)

C++ به شما اجازه می‌دهد که از حلقه‌ها به صورت تو در تو استفاده کنید. اگر یک حلقه در داخل حلقه دیگر قرار بگیرد، به آن حلقه تو در تو گفته می‌شود. در این نوع حلقه‌ها، به ازای اجرای یک بار حلقه بیرونی، حلقه داخلی به طور کامل اجرا می‌شود. در زیر نحوه ایجاد حلقه تو در تو آمده است:

```
for (init; condition; increment)
{
    for (init; condition; increment)
    {
        //statement(s);
    }
    //statement(s);
}
```

نکته ای که در مورد حلقه‌های تو در تو وجود دارد این است که می‌توان از یک نوع حلقه در داخل نوع دیگر استفاده کرد. مثلاً می‌توان از حلقه for در داخل حلقه while استفاده نمود. در مثال زیر نحوه استفاده از این حلقه‌ها ذکر شده است. فرض کنید که می‌خواهید یک مستطیل با ۳ سطر و ۵ ستون ایجاد کنید:

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6      for (int i = 1; i <= 4; i++)
7      {
8         for (int j = 1; j <= 5; j++)
9         {
10                cout << " * ";
11        }
12        cout << endl;
13     }
14
15  }
```

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

تمرین : برنامه ای بنویسید که جدول ضرب ۵ در ۵ در خروجی چاپ کند

```
#include <iostream.h>
int main( )
{

  for (int x=1;x<=5;x++)
   {
     for (int y=1;y<=5;y++)
        cout <<x*y<<"\t";
     cout<<endl;
   }

  return 0;
}
```

خروجی برنامه به صورت زیر خواهد بود:

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

خارج شدن از حلقه با استفاده از break و continue

گاهی اوقات با وجود درست بودن شرط می‌خواهیم حلقه متوقف شود. سؤال اینجاست که چطور این کار را انجام دهید؟ با استفاده از کلمه کلیدی **break** حلقه را متوقف کرده و با استفاده از کلمه کلیدی **continue** می‌توان بخشی از حلقه را رد کرد و به مرحله بعد رفت.

برنامه زیر نحوه استفاده از continue و break را نشان می‌دهد:

```cpp
#include <iostream>

using namespace std;

int main()
{
   cout << "Demonstrating the use of break" << endl;

   for (int x = 1; x < 10; x++)
   {
      if (x == 5)
         break;

      cout << "Number " << x << endl;
   }

   cout << endl;

   cout << "Demonstrating the use of continue." << endl;

   for (int x = 1; x < 10; x++)
   {
      if (x == 5)
         continue;

      cout << "Number " << x << endl;
   }
}
```

```
Demonstrating the use of break.

Number 1
Number 2
Number 3
Number 4

Demonstrating the use of continue.

Number 1
Number 2
Number 3
Number 4
Number 6
Number 7
Number 8
Number 9
```

تمرین : خروجی برنامه زیر را مشخص نمایید

```
#include <iostream.h>

int main( )
{
  int n=0, sum=0;
  while (n < 20)
  {
    ++n; // n = n + 1;

    if (n==10) continue;
    sum += n; // sum = sum + n;
  }
  cout << "1+2+ ...(except 10)...+20=" <<sum << endl;

  return 0;
}
```