فصل هفتم

دستورات تکرار (حلقه‌ها)

در بعضی از برنامه‌های کاربردی باید یک عمل چندین بار تکرار شود. مثلاً اگر بخواهیم میانگین یا معدل نمرات درس زبان انگلیسی یک کلاس را محاسبه کنیم، باید نمرات تمام دانش‌آموزان کلاس را از ورودی دریافت کرده و با یکدیگر جمع کنیم. در این مثال، عمل دریافت نمره از ورودی و عمل جمع زدن نمره‌ها، به تعداد دانش‌آموزان کلاس، باید تکرار گردد. نوشتن چنین برنامه‌هایی با دستورات تکراری، خسته کننده و طولانی و گاهی غیرممکن خواهد بود. در زبان‌های برنامه‌نویسی از جمله زبان C#، دستورات ایجاد حلقه، برای کوتاه کردن تعداد دستورات برنامه، پیش‌بینی شده‌اند. به‌وسیلهٔ این دستورات، برنامه‌نویس می‌تواند، عملیات و پردازش‌های تکرار شونده را فقط یک بار بنویسد و کامپیوتر آنها را به دفعات، تکرار کند. در این فصل با انواع دستورات حلقه و کاربرد آنها، آشنا می‌شویم.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند :

۱ــ کاربرد حلقه در برنامه را توضیح دهد.

۲ــ دستورات ایجاد حلقه را نام ببرد و تفاوت هر یک را بیان کند.

۳ــ عملکرد و کاربرد دستور حلقه while را توضیح دهد.

۴ــ عملکرد و کاربرد دستور حلقه for را توضیح دهد.

۵ــ برنامه‌های کاربردی با حلقه تکرار بنویسد.

۱ــ۷ــ دستورات تکرار شرطی

فرض کنید، می‌خواهیم برنامه‌ای بنویسیم که فقط، افراد خاصی مجاز به استفاده از آن باشند. بدین منظور، در ابتدای برنامه، نام کاربری و کلمه عبور را سؤال می‌کنیم. اگر کاربر توانست اطلاعات خواسته شده را به طور صحیح وارد کند، به قسمت‌های بعدی برنامه هدایت می‌شود و در غیر این‌صورت، مجددا نام کاربری و کلمه عبور درخواست می‌شود.

در چنین برنامه‌هایی، عمل دریافت اطلاعات، ممکن است تکرار گردد. تکرار دستورات یک برنامه، بسته به نوع الگوریتم آن، می‌تواند با دفعات معین و یا نامعین باشد. زمانی که تعداد دفعات نامشخص است، توقف و یا تکرار بستگی به برقراری یک شرط دارد. در این گونه موارد از دستورات حلقه شرطی مانند while یا do_while استفاده می‌کنیم. اگر تعداد دفعات تکرار مشخص باشد، مثلاً حداکثر ۳ بار نام کاربری و کلمه عبور دریافت گردد، از دستور حلقه معین for استفاده می‌شود.

۱ــ۱ــ۷ ــ دستور حلقه شرطی while : ساختار کلی دستور while، در زیر نشان داده شده است :

```
while (عبارت منطقی)
    دستور ;
```

دستور while از سه بخش تشکیل شده است :

۱ــ کلمه رزرو شده while

۲ــ عبارت منطقی در داخل پرانتز

۳ــ دستوری که در صورت درست بودنِ نتیجه عبارت، اجرا خواهد شد.

مثال ۱ــ۷ : نمونه‌ای از به‌کار گیری دستور while چنین است :

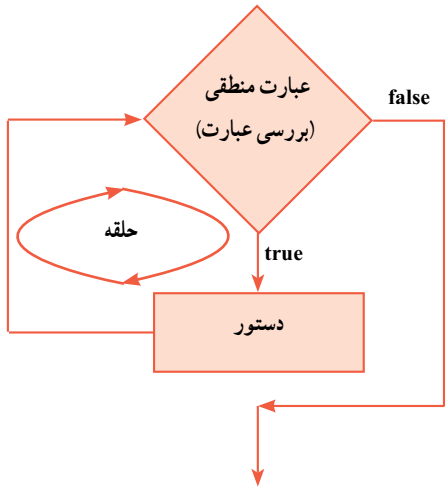
```
int x  1;
while (x < 100)
    Console.WriteLine("x "   x   );
```

قطعه برنامه ۱ــ۷ــ مثالی از یک حلقه

سؤال: به نظر شما خروجی این دستورات چیست ؟ (چه اعدادی روی صفحه نمایش، نشان داده می‌شود؟)

نکته

به علامت نقطه ویرگول در دستور while توجه کنید. بعد از علامت پرانتز علامت نقطه ویرگول نگذارید، زیرا دستور while هنوز تمام نشده است. علامت نقطه ویرگول باید در انتهای دستور نوشته شود.



فلوچارت ۱ــ۷ــ دستور while

هنگامی که کامپیوتر در حال اجرای برنامه است، با رسیدن به دستور while، ابتدا مقدار عبارت را بررسی می‌کند. در صورتی که مقدار عبارت true باشد، دستور (یا بلاک) نوشته شده بعد از while، اجرا می‌شود. پس از آن، دوباره مقدار عبارت محاسبه می‌شود و تا زمانی که ارزش آن true باشد، دستورمذکور، اجرا خواهد شد. در این حالت می‌گوییم **حلقه**[1] ایجاد شده است (فلوچارت ۱ــ۷).

دستور یا دستوراتی که مکرر اجرا می‌گردند در بدنه حلقه[2] قرار دارد. اگر در ارزیابی عبارت، مقدار false حاصل شود، دستورات بدنه حلقه دیگر اجرا نخواهند شد. برنامه از حلقه خارج می‌شود و دستورات بعدی اجرا می‌شوند.

نکته

اگر بخواهید بیش از یک دستور تکرار گردد، باید آنها را به‌صورت یک بلاک بنویسید. یعنی آنها را در داخل علامت‌های آکولاد باز و بسته قرار دهید.

کار در کارگاه ۱

۱ــ مثال ۱ــ۷ را به صورت یک برنامه کامل در VS تایپ و اجرا نمایید.

۲ــ اعداد حلقه را طوری تغییر دهید تا خروجی اعداد دو رقمی شود.

۳ــ اعداد حلقه را طوری تغییر دهید تا خروجی اعداد سه رقمی و به صورت نزولی شود.

۱ــ Loop ۲ــ Loop Body

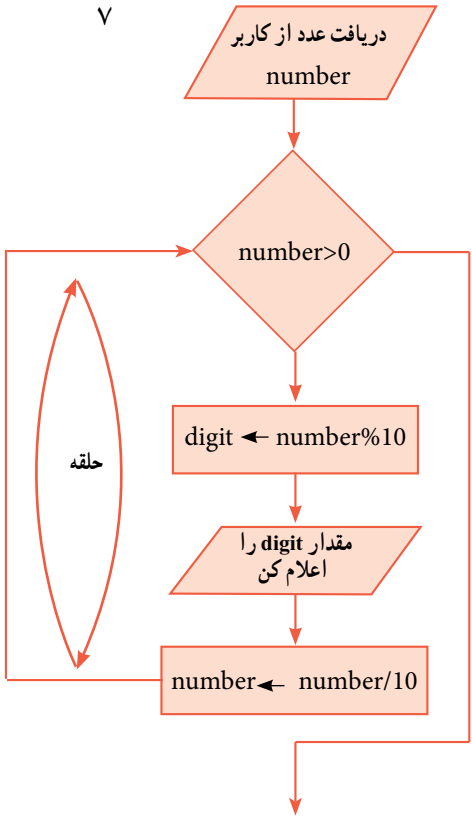
مثال ۲ــ۷ : می‌خواهیم برنامه‌ای بنویسیم که ارقام یک عدد را جدا نموده و آن‌ها را نمایش دهد.

الگوریتم یا روش انجام کار : با توجه به آن که رقم یکان هر عدد، باقیمانده تقسیم آن عدد صحیح بر ۱۰ است، کافی است عدد دریافتی را بر ۱۰ تقسیم و باقیمانده آن را نمایش دهیم. به عنوان مثال اگر عدد دریافتی ۵۷۶ باشد باقیمانده تقسیم آن بر عدد ۱۰، عدد ۶ است که رقم یکان عدد است.

$$\begin{array}{r|l} ۵۷۶ & ۱۰ \\ \hline ۵۷۰ & ۵۷ \\ \hline ۷ & \end{array}$$

اگر دوباره خارج قسمت بدست آمده یعنی ۵۷ را بر عدد ۱۰ تقسیم کنیم، خواهیم داشت :

$$\begin{array}{r|l} ۵۷ & ۱۰ \\ \hline ۵۰ & ۵ \\ \hline ۷ & \end{array}$$



فلوچارت ۲ــ۷ــ جدا کردن ارقام عدد

اگر به باقیمانده تقسیم بالا توجه کنید، متوجه می‌شوید که عدد ۷، رقم ده گان عدد دریافتی است. به همین ترتیب ادامه می‌دهیم و عمل تقسیم را تکرار کرده و باقیمانده تقسیم را به‌دست می‌آوریم.

$$\begin{array}{r|l} ۵ & ۱۰ \\ \hline ۰ & ۰ \\ \hline ۵ & \end{array}$$

باقیمانده تقسیم بالا را در نظر بگیرید. در اینجا توانستیم آخرین رقم عدد یعنی ۵ را نیز جدا کنیم.

با دقت در عملیات فوق متوجه می‌شویم که عمل تقسیم، عملی تکراری است و تا زمانی انجام می‌شود که مقسوم آن بزرگتر از صفر باشد (فلوچارت ۲ــ۷).

مطابق با فلوچارت شکل ۲ــ۷، برنامه را می‌نویسیم :

```
class Numbers
{
    static void Main()
    {
        int number, digit ;
        string input;
        Console.Write("Enter a number:  ");
        input   Console.ReadLine();
        number   int.Parse(input);
        while (number > 0)
        {
            digit   number % 10;
            Console.WriteLine(digit);
            number /   10;
        }
        Console.WriteLine("Press any key to continue…");
        Console.ReadKey();
    }
}
```

برنامه ۲ــ۷ــ جدا کردن ارقام یک عدد صحیح

سؤال: در برنامه ۲ــ۷، یک بلاک شامل سه دستور، در داخل حلقه قرار دارد که تکرار می‌شود. آن بلاک و دستورهای داخل آن را مشخص کنید.

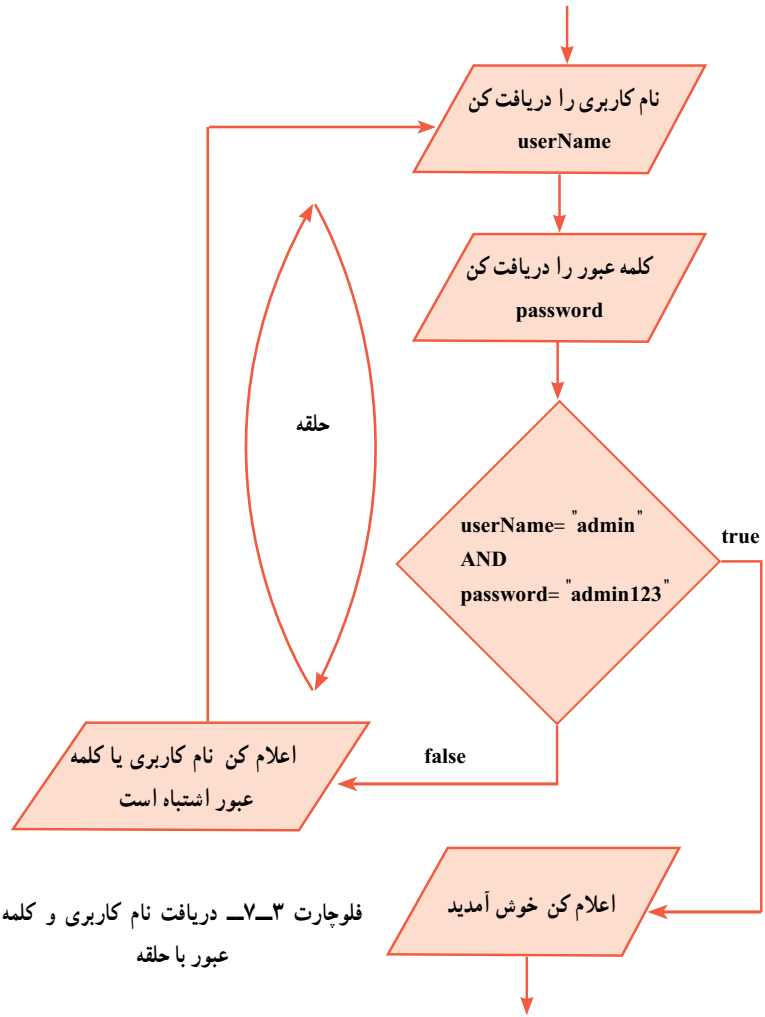
سؤال: برنامه را برای عدد ۳۸۵ در جدول، Trace کنید.

input	number	digit	خروجی
385			
۱۴۵			

سؤال: آیا ممکن است که دستورات داخل حلقه، اصلا اجرا نشوند؟ به ازای چه مقادیری این اتفاق می‌افتد؟

مثال ۳ــ۷ : می‌خواهیم برنامه‌ای بنویسیم که نام کاربری و رمز عبور را سؤال نماید و اگر کاربر اطلاعات خواسته شده را به درستی وارد نکرد، دوباره سؤال شود.

الگوریتم یا روش انجام کار : در فصل ششم در مثال ۱۰ــ۶، با روش دریافت و بررسی نام کاربری و کلمه عبور آشنا شدید. در این مثال از حلقه برای تکرار عملیات استفاده می‌کنیم. در صورتی که کاربر اطلاعات را به‌طور صحیح وارد نکرد، باید دوباره عمل دریافت و بررسی اطلاعات تکرار شود. فلوچارت ۳ــ۷، حلقه و عملیات تکراری را نشان می‌دهد.



فلوچارت ۳ــ۷ــ دریافت نام کاربری و کلمه عبور با حلقه

اگر فلوچارت ۳ــ۷ را با دقت بررسی کنید، متوجه می‌شوید که در این مثال، ابتدا دستورات داخل حلقه اجرا می‌شوند و سپس شرط ادامه تکرار، بررسی می‌شود. در حالی که در مثال قبلی، ابتدا شرط بررسی می‌شد و سپس در صورت برقراری شرط، دستورات داخل حلقه اجرا می‌شد. در زبان برنامه‌نویسی C#، دستور حلقه do_while برای این گونه مسایل در نظر گرفته شده است، که در این قسمت به شرح آن می‌پردازیم.

۲ــ۱ــ۷ دستور حلقه شرطی do_while : شکل کلی دستور do_while به‌صورت زیر است :

```
do
    دستور;
while (عبارت منطقی) ;
```

دستور do_while از چهار بخش تشکیل شده است :

۱ــ کلمه رزرو شده do

۲ــ دستور داخل حلقه

۳ــ کلمه رزرو شده while

۴ــ عبارت منطقی داخل پرانتز، که در صورت درست بودن آن، دستور داخل حلقه تکرار می‌شود.

مثال ۴ــ۷ : نمونه‌ای از به‌کارگیری دستور do_while چنین است :

```
int x  1;
do
    Console.WriteLine("x  "   x   );
while (x < 100);
```

قطعه برنامه ۳ــ۷ــ مثالی از یک حلقه

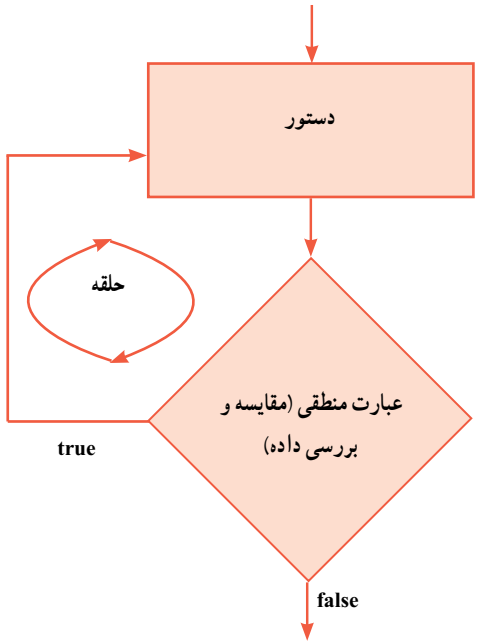
سؤال: به نظر شما خروجی این دستورات چیست؟ (چه اعدادی روی صفحه نمایش، نشان داده می‌شود؟)

نکته

اگر بخواهید بیش از یک دستور در حلقه قرار گیرد، باید آنها را در یک بلاک قرار دهید.

به محل نوشتن علامت ; در دستور do_while توجه کنید. این علامت بعد از عبارت منطقی باید نوشته شود.

کامپیوتر با رسیدن به دستور do_while، ابتدا دستور داخل حلقه را اجرا می‌کند که از کلمه do شروع می‌شود و سپس با رسیدن به کلمه while، مقدار عبارت منطقی را ارزیابی می‌نماید. اگر حاصل عبارت true باشد، آنگاه به قسمت do برمی‌گردد و دستور بدنه حلقه اجرا می‌شود. تا زمانی که حاصل عبارت true است حلقه تکرار می‌شود. اگر حاصل ارزیابی عبارت false شود، دیگر به کلمه do برنمی‌گردد و کنترل برنامه به خط بعد از while واگذار شده و دستورات دیگرِ برنامه اجرا می‌شوند. فلوچارت ۴ـ۷، دستور do_while را نشان می‌دهد.



فلوچارت ۴ـ۷ـ دستور do_while

کار در کارگاه ۲

برنامه مربوط به فلوچارت ۳ـ۷ (دریافت نام کاربری و کلمه عبور)، را با استفاده از حلقه do_while بنویسید.

```
class LoginLoop
{
    static void Main(string[] args)
    {
        string userName, password;
        bool loginFlag;
        do
        {
            Console.Write("Enter username: ");
            userName = Console.ReadLine();
            Console.Write("Enter password: ");
            password = Console.ReadLine();
            if ((userName == "admin") && (password == "admin123"))
                loginFlag = true;
            else
            {
                loginFlag = false;
                Console.WriteLine("Wrong username or password!. Try again.");
            }
        } while (!loginFlag);
        Console.WriteLine("Welcome Admin.");
        Console.WriteLine("Press any key to continue...");
        Console.ReadKey();
    }
}
```

حلقه do...while

برنامه ۴ــ۷ــ دریافت نام کاربری و رمز عبور در داخل حلقه

در برنامه ۴ــ۷، بیش از یک دستور در داخل حلقه قرار دارد، بنابراین ازعلامت‌های آکولاد باز و بسته برای ایجاد یک بلاک استفاده شده است که بین کلمات do و while قرار دارند. در داخل بلاک ابتدا نام کاربری و کلمه عبور دریافت شده است و سپس درستی آنها توسط دستور if، بررسی شده است. برای کنترل حلقه (یا شرط تکرار حلقه) از یک متغیر منطقی به نام loginFlag استفاده شده است. اگر کاربر اطلاعات نام کاربری و کلمه عبور را صحیح وارد کند، در این متغیر مقدار true قرار می‌گیرد. اما اگر کاربر اطلاعات نادرست وارد کند، در این متغیر مقدار false قرار می‌گیرد.

به عبارت منطقی کنترل حلقه که پس از کلمه while نوشته شده است دقت کنید :

```
} while (!loginFlag);
```

در عبارت منطقی، از عملگر نقیض استفاده شده است. بنابراین تا زمانی که مقدار متغیر loginFlag برابر false است، حلقه تکرار می‌گردد. هر گاه کاربر، اطلاعات صحیح را وارد کند در متغیر loginFlag مقدار true قرار گرفته و در نتیجه حاصل عبارت منطقی false شده و حلقه دیگر تکرار نمی‌شود. در نتیجه دستور بعد از While اجرا می‌شود که نمایش یک پیام خوشامد گویی است.

```
Console.WriteLine("Welcome Admin.");
```

سؤال: چرا در قسمت else از علامت‌های آکولاد باز و بسته استفاده شده است، اما در قسمت if، چنین نیست؟

مثال ۷ــ۷ : می‌خواهیم یک بازی حدس عدد، ایجاد کنیم. این بازی بین دو بازیکن به شرح زیر صورت می‌گیرد. بازیکن اول عددی را برای خود در نظر می‌گیرد و بازیکن دوم باید آن عدد را حدس بزند. بازیکن اول در طول بازی، راهنمایی لازم را در اختیار بازیکن دوم قرار می‌دهد تا عدد بالاتر یا پایین تری را حدس بزند.

الگوریتم یا روش انجام کار : با توجه به شرح بازی، ابتدا عدد مورد نظر بازیکن اول را سؤال کرده و در یک متغیر (number) ذخیره می‌کنیم. سپس از بازیکن دوم می‌خواهیم تا عددی که بازیکن اول وارد کرده است را حدس بزند. عدد دریافتی از بازیکن دوم (guess)، باید با عدد مورد نظر بازیکن اول مقایسه شود که در این صورت سه حالت رخ می‌دهد :

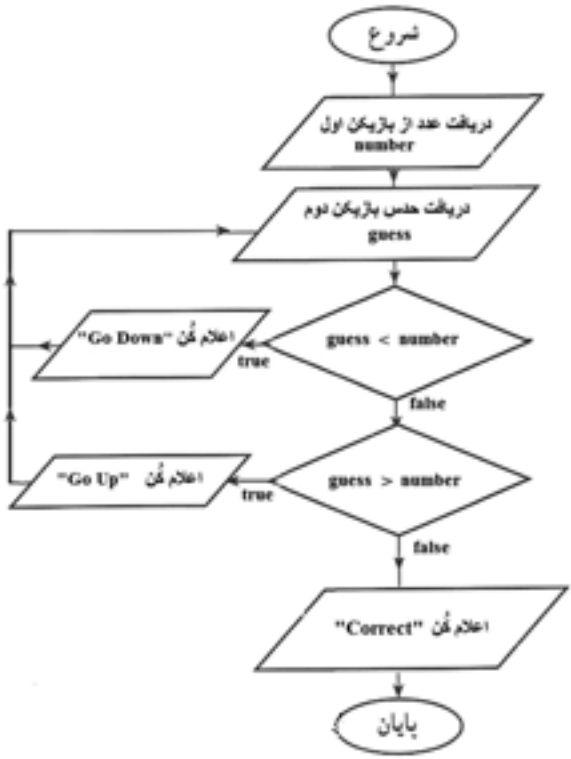
حالت اول : guess number است که در این صورت پیام «آفرین درست حدس زدید» اعلان شود.

حالت دوم : guess < number است که در این صورت پیام «برو بالا» اعلان شود.

حالت سوم : guess > number است که در این صورت پیام «برو پایین» اعلان شود.

تا زمانی که بازیکن دوم، عدد را درست حدس نزده است، عملیات دریافت عدد از بازیکن دوم و مقایسه آن باید تکرار شود. بنابراین در این برنامه نیاز به یک حلقه می‌باشد. از آن جا که عمل حدس زدن عدد، حداقل یک بار به وسیلهٔ بازیکن دوم انجام می‌شود، به نظر شما از چه دستور حلقه باید استفاده کنیم؟

فلوچارت ۵ــ۷، لازم برای انجام این بازی را نشان می‌دهد. آن را به ازای ورودی‌های مختلف دنبال کنید.



فلوچارت ۵ــ۷ــ بازی حدس عدد

بر اساس فلوچارت فوق، برنامه ۵ــ۷ را می‌نویسیم.

```
class GussTheNumber
{
    static void Main(string[] args)
    {
        string input;
        int number, guess;
        Console.Write("Player 1: Think a number (1100_): ");
        input   Console.ReadLine();
        number   int.Parse(input);
        Console.Clear(); // clear console screen
        do
        {
            Console.Write("Player 2: Guess the number (1100_): ");
            input   Console.ReadLine();
            guess   int.Parse(input);
            if (guess < number)
                Console.WriteLine("Incorrect, Go Up.");
            else
                if (guess > number)
                    Console.WriteLine("Incorrect, Go Down.");
                else
                    Console.WriteLine("Well done!, it's correct.");
        } while (guess !  number);
        Console.WriteLine("Press any key to continue...");
        Console.ReadKey();
    }
}
```

حلقه do_while

در برنامه ۵ــ۷، از حلقه do_while استفاده شده است. به عبارت منطقی در دستور while توجه کنید :

```
} while (guess ! number);
```

اگر حدس بازیکن دوم، مخالف عدد مورد نظر بازیکن اول باشد، نتیجه عبارت true است و حلقه تکرار می‌شود. تنها در صورتی که هر دو عدد با یکدیگر برابر باشند، نتیجه عبارت false خواهد شد و حلقه قطع شده و دستور بعد از while اجرا می‌شود. توجه کنید که چون دستورات داخل حلقه بیش از یک دستور است، در داخل یک بلاک قرار گرفته اند.

گسترش برنامه : معمولاً برنامه‌ای که نوشته می‌شود، ایده‌آل و کامل نیست. امکان اضافه کردن ویژگی و قابلیت‌های جدید در هر برنامه وجود دارد. در برنامه ۵ــ۷ نیز، می‌توان امکاناتی را اضافه کرد. به‌عنوان مثال، قابلیت امتیازدهی را به برنامه اضافه می‌کنیم. این امتیاز باید متناسب با تعداد دفعاتی باشد، که بازیکن دوم تلاش می‌کند تا عدد مورد نظر بازیکن اول را پیدا کند.

در قسمت کار در کارگاه، در انتهای این فصل، چنین امکانی را به برنامه ۵ــ۷ اضافه می‌کنیم.

۲ــ۷ــ دستور حلقه for

در مقدمه این فصل، مثالی در مورد محاسبه میانگین نمرات درسی یک کلاس بیان شد که در آن، عملیات دریافت نمرات و محاسبه مجموع آنها، باید به تعداد دانش آموزان یک کلاس تکرار شود. در چنین برنامه‌هایی که در آن تعداد تکرار دستورات معین است، بهتر است از دستور حلقه for استفاده کنیم که برای این منظور در زبان C# پیش‌بینی شده است. برای آشنایی با این دستور با یک مثال ساده جهت نمایش اعداد ۱ تا ۱۰ شروع می‌کنیم.

مثال ۶ــ۷ : می‌خواهیم اعداد طبیعی از ۱ تا ۱۰ را روی صفحه نمایش، نشان دهیم. از دستور for به صورت زیر استفاده می‌کنیم :

```
for (int i   1 ; i<  10 ; i    )
    Console.WriteLine( i);
```

قطعه برنامه ۶ــ۷ــ استفاده از حلقه for برای نمایش اعداد طبیعی ۱ تا ۱۰

دستور ()WriteLine در این مثال در داخل حلقه قرار دارد که در هر بار تکرار حلقه، مقدار متغیر i روی صفحه نمایش، نشان داده می‌شود. مقدار متغیر i چقدر است؟

برای پاسخ به این سؤال، به داخل پرانتز، در جلوی دستور for، توجه کنید. سه قسمت در داخل پرانتز، قابل تشخیص است. در قسمت اول، یک متغیر از نوع صحیح به نام i با مقدار اولیهٔ ۱ تعریف شده است. قسمت دوم، یک عبارت منطقی (i < 10) است و قسمت سوم نیز یک دستور انتساب افزایشی (i) است.

جزییات اجرای دستورات بالا چنین است : در دستور for، ابتدا عدد ۱ در متغیر i قرار می‌گیرد و سپس عبارت منطقی محاسبه می‌شود و چون (i< 10) است، نتیجه عبارت true است. بنابراین دستور داخل حلقه اجرا می‌شود و در نتیجه عدد ۱ در روی صفحه نمایش داده می‌شود. سپس دستور انتساب افزایشی انجام می‌شود یعنی مقدار متغیر به اندازه یک واحد افزایش می‌یابد.

دوباره عبارت منطقی محاسبه می‌شود و چون هنوز نتیجه عبارت درست است (۲< ۱۰) در نتیجه دستور داخل حلقه اجرا شده و عدد ۲ بر روی صفحه نشان داده می‌شود. این عملیات تکرار می‌شود تا زمانی‌که نتیجه عبارت نادرست شود یعنی (۱۱< ۱۰) که در این‌صورت حلقه قطع می‌شود. بنابراین اعداد ۱ تا ۱۰ روی صفحه نمایش، نشان داده می‌شود.

با توجه به مثال، شکل کلی دستور for چنین است :

```
for ( نام متغیر    مقدار اولیه ; عبارت منطقی ; تغییر مقدار متغیر )
    دستور ;
```

کلمه for، یک کلمه رزرو شده است. متغیری که در حلقه for استفاده می‌شود، هر نام مجازی می‌تواند داشته باشد. این متغیر به نام شمارنده[۱] معروف می‌باشد. چون نقشِ شمارش تعداد تکرار حلقه را به عهده دارد. شمارش می‌تواند به‌صورت صعودی یا نزولی انجام شود. بدیهی است در حالت نزولی، تغییر مقدار متغیر باید به‌صورت کاهشی باشد.

شکل‌های دیگر حلقه مثال ۶ــ۷ در زیر آورده شده است.

```
for (int i    1 ; i<   10 ; Console.WriteLine( i) , i    );
```

سؤال: خروجی این دستور را بررسی کنید.

```
for (int i    1 ; i<   10 ; i    , Console.WriteLine( i) );
```

سؤال: خروجی این دستور را بررسی کنید.

۱ــ Counter

مثال ۷ــ۷ : در دستور زیر از یک حلقه نزولی استفاده شده است. به هر سه قسمت داخل پرانتز دقت کنید.

```
for (int i    100 ; i> 1 ; i      )
    Console.WriteLine( i);
```

قطعه برنامه ۷ــ۷ــ استفاده از حلقه for کاهشی

سؤال: به نظر شما خروجی این دستورات چیست ؟ (چه اعدادی روی صفحه نمایش، نشان داده می‌شود؟)

کاربرد break در ساختار for : قطعه برنامه زیر را در نظر بگیرید :

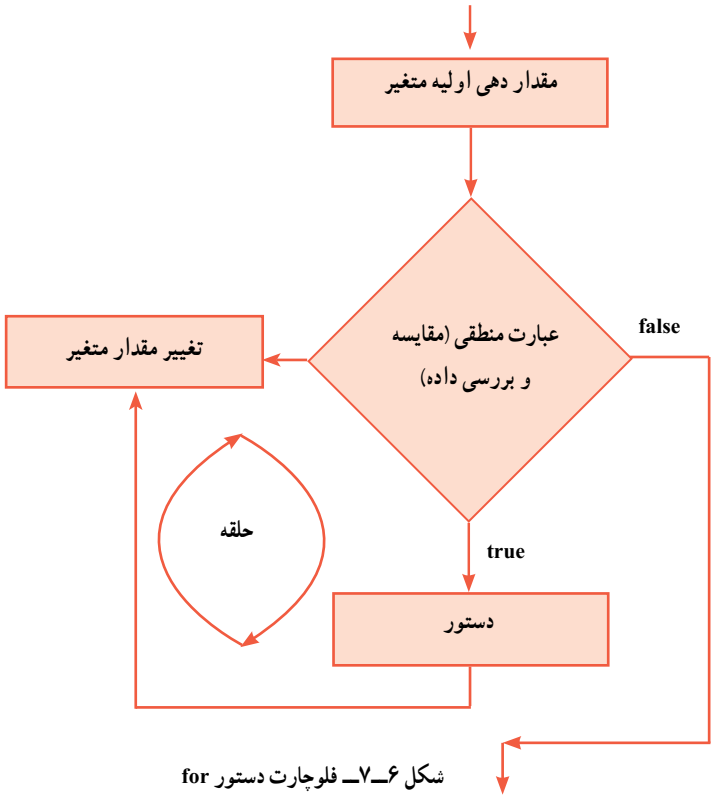
```
int i;
for (i    20 ; i>   1 ; i      )
{
    if (i    5)
       break;
    Console.WriteLine(i);
}
Console.WriteLine("broke out of loop at i of "    i);
```

در این قطعه برنامه از دستور break برای خروج از حلقه در صورت برقراری شرط (i 5) استفاده شده است.این حلقه اعداد ۲۰ تا ۶ را به صورت نزولی چاپ می‌کند و با رسیدن به عدد ۵ توسط دستور break حلقه خاتمه می‌یابد.

توجه داشته باشید، که هیچ کدام از سه قسمت داخل پرانتز، در دستور for اجباری نیستند. حتی دستور for، می‌تواند به‌صورت زیر نوشته شود که در این صورت، یک حلقه تمام نشدنی و بی‌نهایت ایجاد می‌شود.

```
for ( ; ; )
    Console.WriteLine("Infinite Loop! ");
```

شکل ۶ــ۷، فلوچارت دستور for را نشان می‌دهد. در این فلوچارت، دستور ایجاد و مقداردهی اولیه متغیر، در داخل حلقه قرار ندارد و تنها یک بار در ابتدا اجرا می‌شود.



شکل ۶ــ۷ــ فلوچارت دستور for

نکته

به علامت (;) در دستور for توجه کنید. بعد از علامت پرانتز علامت(;) نگذارید، زیرا دستور for هنوز تمام نشده است. علامت ; باید در انتهای دستور نوشته شود.

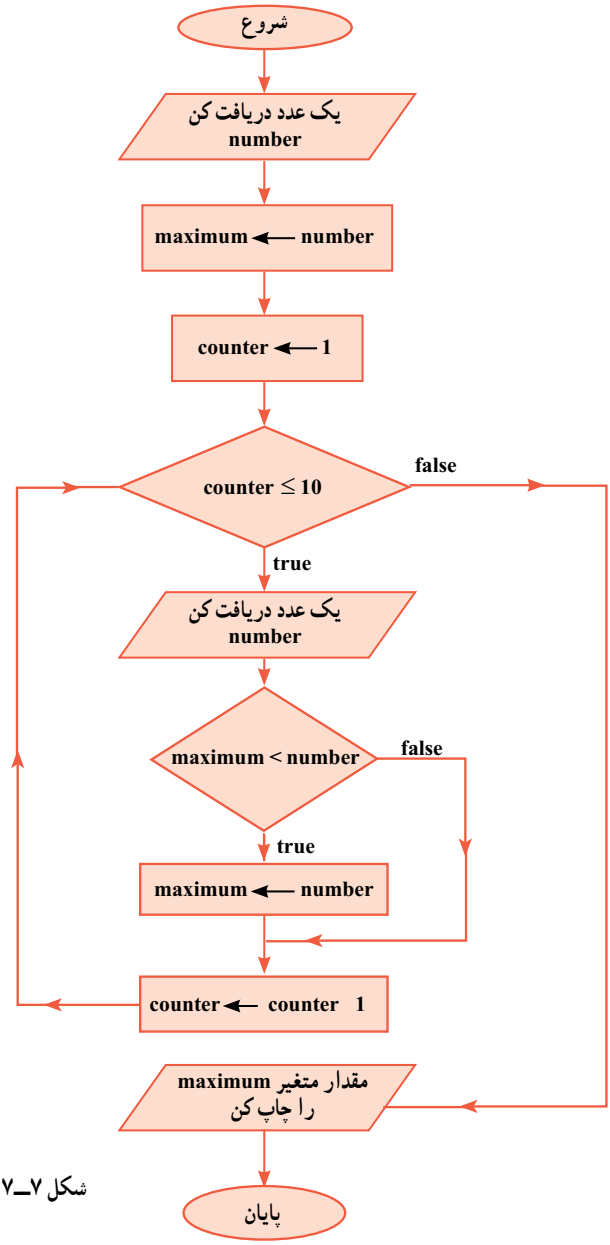
for (نام متغیر مقدار اولیه ; عبارت منطقی ; تغییر مقدار متغیر)

دستور ;

مثال ۸ــ۷ : می‌خواهیم ده عدد از ورودی دریافت کرده، بزرگ‌ترین آن را تشخیص داده و نمایش دهیم.

الگوریتم یا روش انجام کار : برای پیدا کردن بزرگترین عدد، از روشی که در فصل ششم در مثال ۳ــ۶، توضیح داده شد، استفاده می‌کنیم. شکل ۷ــ۷، فلوچارت پیدا کردن بزرگترین عدد، از بین ده عدد را نشان می‌دهد. در ابتدای این فلوچارت، اولین عدد دریافت می‌شود و آن را به عنوان

بزرگترین عدد در متغیر maximum ذخیره می‌کنیم. سپس در داخل حلقه، بقیه اعداد دریافت و با مقدار متغیر maximum مقایسه می‌شوند. هر جا که عدد بزرگتری دریافت شد، در متغیر maximum ذخیره می‌شود. چون حلقه باید به تعداد ۹ بار تکرار گردد، از یک متغیر به نام counter برای شمارش تعداد تکرار حلقه استفاده شده است، که با هر بار تکرار حلقه، یک واحد به آن اضافه می‌شود.



شکل ۷ــ۷ ــ فلوچارت پیدا کردن بزرگ‌ترین عدد

برنامه ۸ــ۷ بر اساس فلوچارت شکل ۷ــ۷ نوشته شده است.

```
class FindMaximum
{
    static void Main(string[] args)
    {
        string input;
        int number, maximum;
        Console.Write("Enter a number: ");
        input  Console.ReadLine();
        maximum  int.Parse(input); // Suppose first number is maximum.

        for (int i  2; i<  10 ; i  )
        {
            Console.Write("Enter a number: ");
            input  Console.ReadLine();
            number  int.Parse(input);
            if (maximum < number) // found bigger number
                maximum  number;

        }
        Console.WriteLine("The maximum number is "  maximum);
        Console.WriteLine("Press any key to continue . . .");
        Console.ReadKey();

    }
}
```

دریافت اولین عدد

دریافت دومین عدد تا دهمین در حلقه

برنامه ۸ــ۷ــ پیدا کردن بزرگترین عدد از بین ده عدد

کار در کارگاه ۳

۱ــ قطعه برنامه ۳ــ۷ را به یک برنامه کامل تبدیل کرده و آن را اجرا کنید.

۲ــ برنامه ۵ــ۷ را نوشته و اجرا کنید. این بازی را با هم گروه خود انجام دهید.

۳ــ به برنامه ۵ــ۷، امکان امتیازدهی را اضافه نمایید. برای این منظور، با استفاده از یک متغیر، تعداد دفعات تلاش بازیکن دوم را شمارش کرده و در انتهای برنامه، محتوای آن را نشان دهید. یک دستور انتساب افزایشی در داخل حلقه می‌تواند تعداد دفعات را شمارش کند.

۴ــ قطعه برنامه‌های ۶ــ۷ و ۷ــ۷ را به یک برنامه کامل تبدیل کرده و آن را اجرا کنید.

۵ــ برنامه ۸ــ۷ را نوشته و اجرا کنید. ده عدد مختلف وارد کرده و نتیجه را مشاهده کنید.

۶ــ با تغییراتی در برنامه ۸ــ۷، کوچکترین عدد را پیدا کنید. نام متغیر minimum را برای کوچک‌ترین عدد استفاده کنید.

۷ــ با استفاده از بند ۸ و ۹، برنامه‌ای بنویسید که هم بزرگترین و هم کوچک‌ترین عدد را از بین ده عدد پیدا کند.

۸ــ به بند ۱۰، دستوری اضافه کنید تا فاصله بین بزرگ‌ترین عدد و کوچک‌ترین عدد را پیدا کند.

تمرینات برنامه‌نویسی فصل هفتم

۱ــ برنامه‌ای بنویسید که اعداد زوج از ۳ تا ۲۱ را به صورت نزولی نشان دهد. خروجی برنامه را روی عدد ۱۶ متوقف کنید

۲ــ برنامه‌ای بنویسید که یک عدد صحیح را دریافت کند و سپس اعداد فرد از ۱ تا آن عدد را چاپ نماید.

۳ــ برنامه‌ای بنویسید که یک عدد صحیح را دریافت کند و مقلوب آن را محاسبه و نمایش دهد. مثلاً اگر عدد ۵۲۹ وارد شد برنامه عدد ۹۲۵ را نمایش دهد.

۴ــ برنامه‌ای بنویسید که نمرات درس انگلیسی دانش آموزان یک کلاس ۱۵ نفری را سؤال نماید و سپس اطلاعات زیر را نمایش دهد :

الف) بالاترین نمره کلاس

ب) کمترین نمره کلاس

پ) فاصله بین کمترین و بیشترین نمره

ت) مجموع نمرات کلاس

ث) میانگین یا معدل نمرات کلاس

۵ـ اجرای دستورات زیر سبب نمایش چه اعدادی می‌شود؟

```
int a   1, b   1 , c  0;
while (a < 30)
{
    Console. WriteLine(a);
    c   a   b;
    a   b;
    b   c;
}
```

۶ـ در یک بازی دو نفره، هفت چوب کبریت قرار دارد. هر یک از بازیکنان می‌توانند در نوبت خود یک یا دو یا حداکثر سه چوب کبریت بردارند. بازیکنی که آخرین چوب کبریت را بردارد، بازنده است. برنامه‌ای بنویسید که این بازی را بین دو بازیکن اجرا کند. در هر مرحله بازی، تعداد چوب کبریت‌های باقیمانده را چاپ کنید. سپس از هر بازیکن، تعداد چوب کبریت‌هایی که مایل است بردارد را سؤال نموده و باقیمانده را چاپ کنید.

پیوست ۱ : نصب Visual Studio

شرکت مایکروسافت نرم‌افزاری به نام Visual Studio ساخته است که برای برنامه‌نویسی استفاده می‌شود این نرم‌افزار شامل مترجم زبان برنامه‌نویسی، یک ویرایشگر[۱] و یک اشکال‌یاب می‌باشد. این نرم‌افزار در چند نسخه با ویژگی‌های مختلف عرضه شده است. همچنین یک نسخه رایگان به نام Visual Studio Express Edition عرضه کرده است که برای شروع کار برنامه‌نویسی مناسب است. دیگر نرم‌افزار رایگان، نرم‌افزار Microsoft . Net Framework است که یک دسته ابزار مختلف به‌صورت فرمان (که در خط فرمان باید تایپ شوند) فراهم می‌کند که برای ترجمه و اجرای برنامه نوشته شده به زبان C# می‌تواند مورد استفاده قرار گیرد. بنابراین شما برای برنامه‌نویسی و اجرای برنامه‌ها دو راه دارید :

۱ــ استفاده از Visual Studio که نیاز به تهیه و نصب نرم‌افزار VS دارد که در ادامه این ضمیمه طریقه نصب آن بیان شده است.

۲ــ استفاده از ابزارهای خط فرمان : نیاز به نصب NET. دارد که در پیوست ۲ طریقه نصب آن را خواهید دید.

پیوست ۲ : نصب Net Framework

برای نصب Net Framework. مراحل زیر را دنبال کنید :

۱ــ از طریق موتور جستجوگری نظیر گوگل، برنامه Net Framework. را جستجو کرده و وارد سایت شرکت مایکروسافت شوید یا به آدرس زیر مراجعه کنید :

http: //www.microsoft.com/net

۲ــ در صفحه مربوط به دانلود برنامه Net Framework. روی نسخه NET. موردنظر جهت دانلود کلیک کنید.

۱ــEd tor

مشکلات احتمالی که ممکن است در ترجمه یا اجرای برنامه پیش آید :

الف) اشکال در ترجمه برنامه

۱ــ اگر در هنگام ترجمه برنامه با پیام خطای زیر روبه‌رو شدید احتمال دارد که برنامه .NET بر روی سیستم شما نصب نشده باشد لذا باید این برنامه را نصب کنید. (به ضمیمه ۲ رجوع شود.)

۲ــ اگر علیرغم نصب برنامه .NET همچنان خطای پیدا نکردن مترجم csc رخ می‌دهد، حتماً در مسیر جستجو (Path)، فولدر حاوی برنامه csc معرفی نشده است. فولدر حاوی متوجم معمولاً در مسیر زیر قرار دارد :

C: / Windows / Microsoft.NET/ Framework/ v4.0.30319

البته بسته به نسخه .NET شماره‌های v4.0.30319 متفاوت می‌باشد. این شماره مربوط به نسخه NET4.5 است.

برای اضافه کردن فولدر مربوطه به مسیر جستجو، عملیات زیر را انجام دهید :

اگر سیستم عامل کامپیوتر شما ویندوز ۷ یا ویستا به بعد است :

۱ــ در میزکار روی آیکن Computer کلیک راست کنید و سپس Properties را انتخاب کنید.

۲ــ در سمت چپ صفحه روی گزینه Advanced System Setting کلیک کنید :

۳ــ در پنجره System Properties روی گزینه Environment Variales کلیک کنید.

۴ــ با کلیک برروی متغیر Path آن را انتخاب کرده و سپس روی کلید Edit کلیک کنید.

۵ــ در پنجره‌ای که ظاهر می‌گردد مسیر برنامه‌های مورد جستجو نوشته شده است مسیر .NET را اضافه کنید.

ب) اشکال در اجرای برنامه

اگر در هنگام ترجمه برنامه با اشکال زیر روبه‌رو شدید ممکن است یکی از دو اشتباه زیر را انجام داده باشید :

۱ــ نام فایل را درست تایپ نکرده‌اید.

۲ــ پسوند فایل برنامه را در بعد از نام فایل ننوشته‌اید.

abstract	as	base	bool
break	byte	case	catch
char	checked	class	const
continue	decimal	default	delegate
do	double	else	enum
event	explicit	extern	false
finally	fixed	float	for
foreach	goto	if	implicit
in	in (generic modifier)	int	interface
internal	is	lock	long
namespace	new	null	object
operator	out	out (generic modifier)	override
params	private	protected	public
readonly	ref	return	sbyte
sealed	short	sizeof	stackalloc
static	string	struct	switch
this	throw	true	try
typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual
void	volatile	while	