



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، خسته‌کننده 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. به وسیله 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:

- ◀ کاربرد حلقه در برنامه را توضیح دهد.
- ◀ دستورات ایجاد حلقه را نام ببرد و تفاوت هر یک را بیان کند.
- ◀ عملکرد و کاربرد دستور حلقه while را توضیح دهد.
- ◀ عملکرد و کاربرد دستور حلقه for را توضیح دهد.
- ◀ برنامه‌های کاربردی را با حلقه تکرار بنویسد.
- ◀ در برنامه‌های خود break و Continue را در صورت لزوم به کار بندد.
- ◀ ضرورت استفاده از حلقه‌های متداخل را توضیح دهد و آنها را در برنامه‌های خود به کار بندد.

۱-۲- 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 این صورت، مجدداً نام کاربری و کلمه عبور درخواست می‌شود.

دستورات تکرار (حلقه‌ها)

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.

۱-۱-۷ — دستور حلقه شرطی `while`: ساختار کلی دستور `while`، در زیر نشان داده

شده است:

(عبارت منطقی) `while`

; دستور

دستور `while` از سه بخش تشکیل شده است:

۱- کلمه رزرو شده `while`

۲- عبارت منطقی در داخل پرانتز

۳- دستوری که در صورت درست بودن نتیجه عبارت، اجرا خواهد شد.

مثال ۱-۷: نمونه‌ای از به کار گیری دستور `while` چنین است:

```
int x=1;
```

```
while (x < 100)
```

```
Console.WriteLine("x=" + x++);
```

قطعه برنامه ۱-۷-مثالی از یک حلقه

❓ سؤال: به نظر شما خروجی این دستورات چیست؟ (چه اعدادی روی صفحه

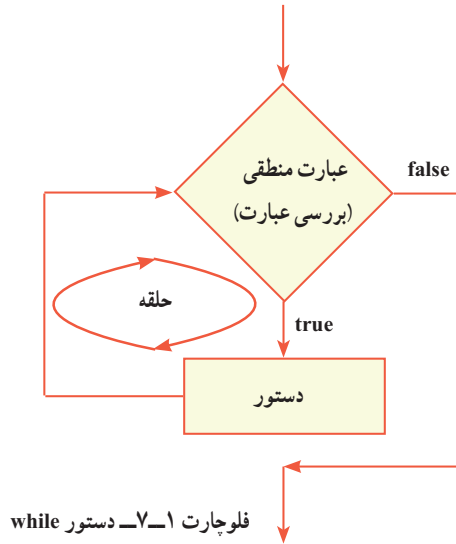
نمایش، نشان داده می‌شود؟)

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.

(عبارت منطقی) `while`
; دستور

نقطه ویرگول ندارد



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، دستور مذکور، اجرا خواهد شد. در این حالت می‌گوییم حلقه^۱ ایجاد شده است (فلوچارت ۷-۱). دستور یا دستوراتی که مکرر اجرا می‌گردند در بدنه حلقه^۲ 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 به صورت یک بلاک بنویسید. یعنی آنها را در داخل علامت‌های آکولاد باز و بسته قرار دهید.

کلیدواژه‌ها

- ۱- مثال ۷-۱ را به صورت یک برنامه کامل در VS تایپ و اجرا نمایید.
- ۲- اعداد حلقه را طوری تغییر دهید تا خروجی اعداد دو رقمی شود.
- ۳- اعداد حلقه را طوری تغییر دهید تا خروجی اعداد سه رقمی و به صورت نزولی شود.

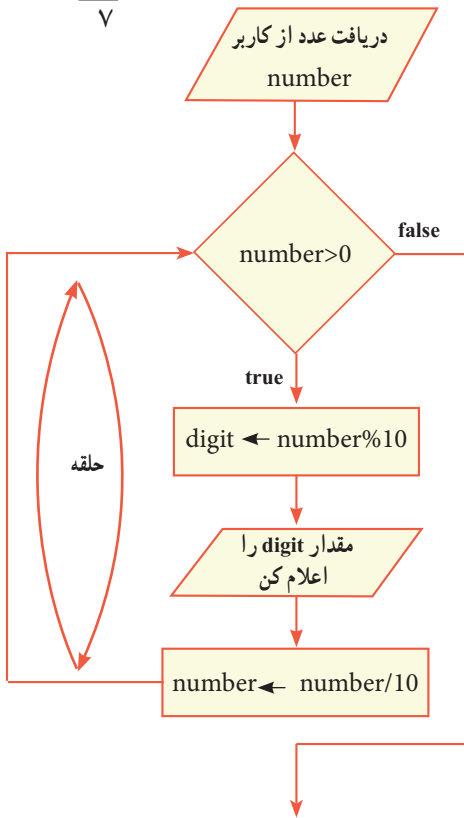
دستورات تکرار (حلقه‌ها)

مثال ۲-۷: 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.

$$\begin{array}{r} 576 \quad | \quad 10 \\ \hline 570 \\ \hline 6 \end{array}$$

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

$$\begin{array}{r} 57 \quad | \quad 10 \\ \hline 50 \\ \hline 7 \end{array}$$



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

$$\begin{array}{r} 5 \quad | \quad 10 \\ \hline 0 \\ \hline 5 \end{array}$$

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 ۲-۷).

فلوچارت ۲-۷- جدا کردن ارقام عدد

مطابق با فلوجارت شکل ۷-۲، برنامه را می‌نویسیم:

```
class Numbers
{
    static void Main()
    {
        int number, digit ;
        string input;
        Console.WriteLine("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 ۷-۲ جدا کردن ارقام یک عدد صحیح

سؤال؟ در برنامه ۷-۲، یک بلاک شامل سه دستور، در داخل حلقه قرار دارد که تکرار می‌شود. بلاک و دستوره‌های داخل آن را مشخص کنید.

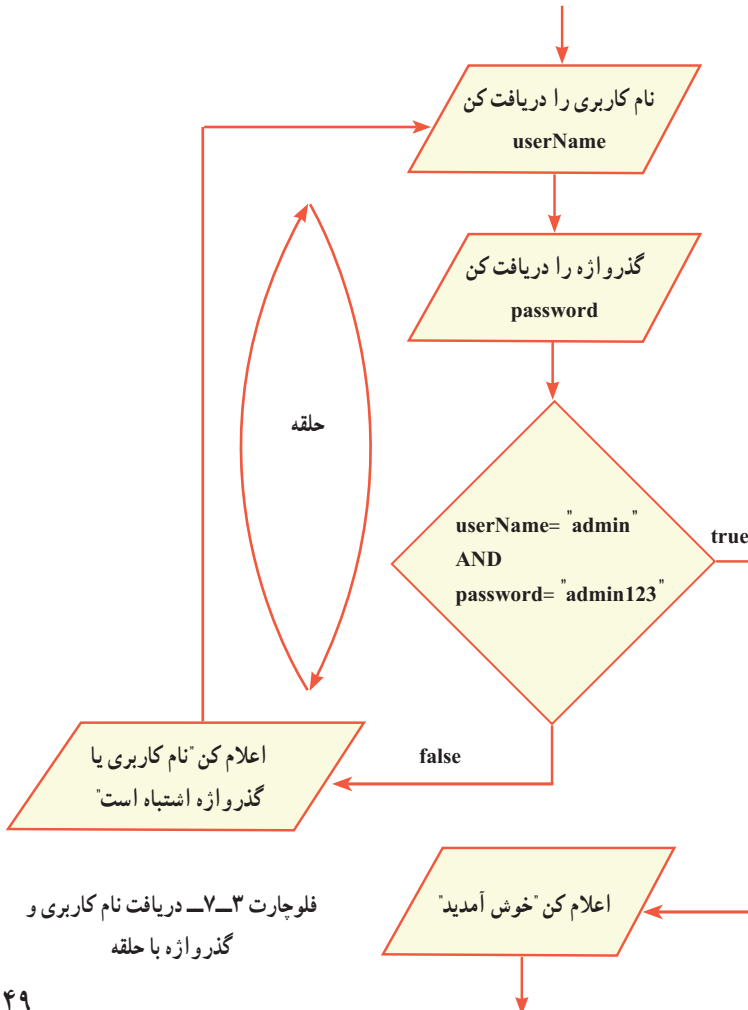
سؤال؟ برنامه را برای عدد ۳۸۵ در جدول، Trace کنید.

input	number	digit	خروجی
385			

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

مثال ۳-۷: 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 ۱۰-۶، باروش دریافت و بررسی نام کاربری و گذرواژه 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. فلوجارت ۳-۷، حلقه و عملیات تکراری را نشان می‌دهد.



فلوجارت ۳-۷— دریافت نام کاربری و گذرواژه با حلقه

اگر فلوجارت ۷-۳ 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.

۲-۱-۷ دستور حلقه شرطی do-while : شکل کلی دستور do-while به صورت زیر است :

do

دستور;

while (عبارت منطقی) ;

دستور do-while از چهار بخش تشکیل شده است :

۱- کلمه رزرو شده do

۲- دستور داخل حلقه

۳- کلمه رزرو شده while

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

مثال ۴-۷ : نمونه‌ای از به کارگیری دستور do-while چنین است :

```
int x=1;
```

```
do
```

```
    Console.WriteLine("x=" + x++);
```

```
while (x < 10);
```

قطعه برنامه ۷-۳-۷ مثالی از یک حلقه

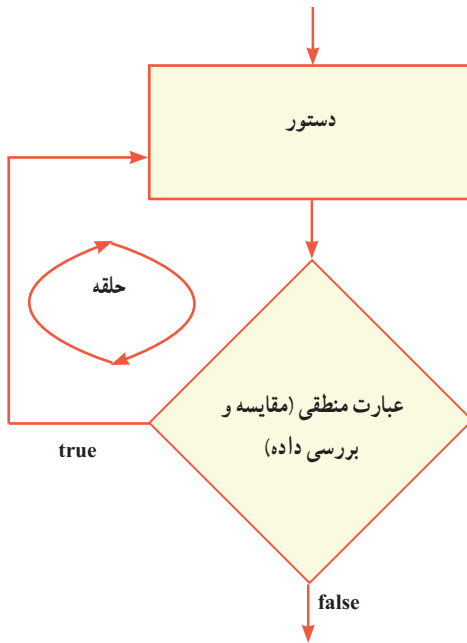
سؤال؟ به نظر شما خروجی این حلقه چیست؟ جدول Trace حلقه را ترسیم کنید.

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 دیگر برنامه اجرا می‌شود. فلوجارت ۷-۴، دستور do_while را نشان می‌دهد.



فلوجارت ۷-۴. دستور do_while

کود دو نگاه ۲

برنامه مربوط به فلوجارت ۷-۳ (دریافت نام کاربری و گذرواژه)، را با استفاده از حلقه do_while

بنویسید.


```

using System;
class LoginLoop
{
    static void Main(string[] args)
    {
        string userName, password;
        bool loginFlag;
        do
        {
            Console.WriteLine("Enter username: ");
            userName = Console.ReadLine();
            Console.WriteLine("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 ۴-۷- دریافت نام کاربری و رمز عبور در داخل حلقه

در برنامه ۴-۷، بیش از یک دستور در داخل حلقه قرار دارد، بنابراین از علامت‌های 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 گذرواژه 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 گذرواژه 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 نفیض 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 خوشامدگویی 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 ۵-۷: 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 پایین‌تری 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:

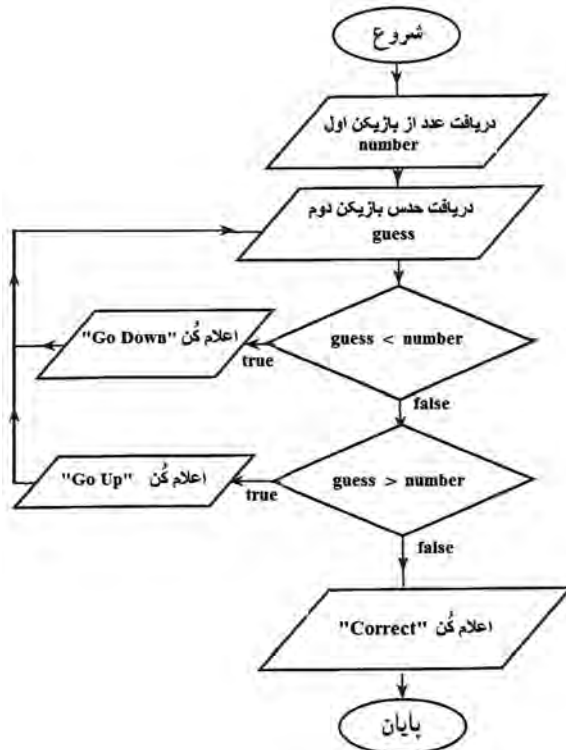
حالت اول : $guess = number$ است که در این صورت پیام «آفرین درست حدس زدید» اعلان شود.

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 می باشد. از آن جا که عمل حدس زدن عدد، حداقل یک بار به وسیله بازیکن دوم انجام می شود، به نظر شما از چه دستور حلقه باید استفاده کنیم؟

فلوچارت ۵-۷، لازم برای انجام این بازی را نشان می دهد. آن را به ازای ورودی های مختلف دنبال کنید.



فلوچارت ۵-۷. بازی حدس عدد

بر اساس فلوچارت فوق، برنامه ۵-۷ را می نویسیم.

```
class GuessTheNumber
{
    static void Main(string[] args)
    {
        string input;
        int number, guess;
        Console.WriteLine("Player 1: Think a number (1-100): ");
        input = Console.ReadLine();
        number = int.Parse(input);
        Console.Clear(); // clear console screen
        do
        {
            Console.WriteLine("Player 2: Guess the number (1-100): ");
            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 ۷-۵، 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 ۷-۵ نیز، می‌توان امکاناتی را اضافه کرد. به عنوان 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 ۷-۵ اضافه می‌کنیم.

۷-۲ 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 ۷-۶ : 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 ۷-۶ 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؟

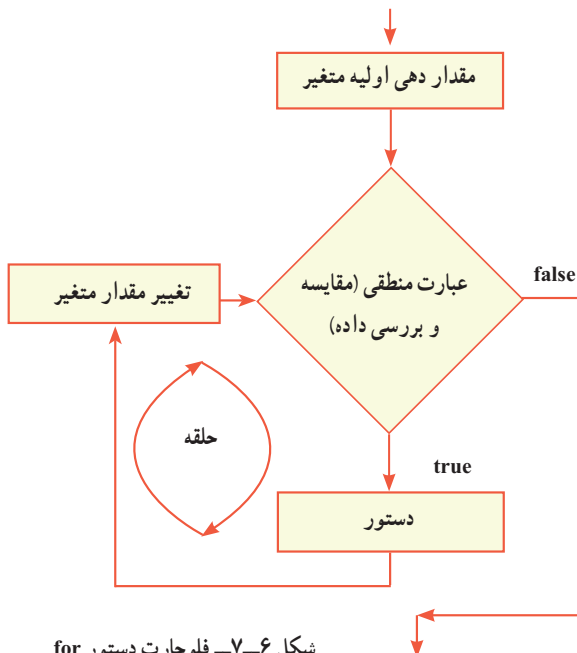
دستورات تکرار (حلقه‌ها)

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

جزئیات اجرای دستورات بالا چنین است: در دستور for، ابتدا عدد ۱ در متغیر i قرار می‌گیرد و سپس عبارت منطقی محاسبه می‌شود و چون ($i \leq 10$) 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 ($2 \leq 10$) در نتیجه دستور داخل حلقه اجرا شده و عدد ۲ بر روی صفحه نشان داده می‌شود. این عملیات تکرار می‌شود تا زمانی که نتیجه عبارت نادرست شود یعنی ($11 \leq 10$) که در این صورت حلقه قطع می‌شود. بنابراین اعداد ۱ تا ۱۰ روی صفحه نمایش، نشان داده می‌شود.

شکل ۶-۷، فلوجارت دستور for را نشان می‌دهد. در این فلوجارت، دستور ایجاد و مقداردهی اولیه متغیر، در داخل حلقه قرار ندارد و تنها یک بار در ابتدا اجرا می‌شود.



شکل ۶-۷. فلوجارت دستور for

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

(تغییر مقدار متغیر؛ عبارت منطقی؛ مقدار اولیه = نام متغیر) for
؛ 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 نقش شمارش تعداد تکرار حلقه را به عهده دارد. شمارش می‌تواند به صورت صعودی یا نزولی انجام شود. بدیهی است در حالت نزولی، تغییر مقدار متغیر باید به صورت کاهشی باشد. شکل‌های دیگر حلقه مثال ۶-۷ 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 و تفاوت آن را با دستور قبل بیان نمایید.

مثال ۷-۷: در دستور زیر از یک حلقه نزولی استفاده شده است. به هر سه قسمت داخل

پراتنزدقت کنید.

```
for (int i = 100 ; i> 1 ; i --)
```

```
Console.WriteLine( i);
```

قطعه برنامه ۷-۷ استفاده از حلقه for کاهشی

سؤال: به نظر شما خروجی این حلقه چیست؟

نکته

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

(تغییر مقدار متغیر؛ عبارت منطقی؛ مقدار اولیه = نام متغیر) for

؛ دستور ←

۱-۲-۷ کاربرد break در ساختار for : جدول Trace مربوط به این حلقه را رسم نمایید.

مثال ۸-۷ : می‌خواهیم برنامه‌ای برای ATM^۱ بنویسیم که سه بار اجازه ورود گذرواژه را به کاربر بدهد و در صورتی که هر سه بار گذرواژه اشتباه وارد شود، کارت توسط دستگاه عابربانک با صدور پیغامی ضبط می‌شود.

الگوریتم یا روش انجام کار : دستگاه عابربانک از شما سه بار گذرواژه ورود می‌گیرد. در صورتی که مقدار ورودی صحیح باشد، شما با پیغام خوش آمدگویی مواجه می‌شوید و با دستور break از حلقه خارج می‌شوید و در صورتی که مقدار ورودی درست نباشد، تا ۳ بار اجازه وارد کردن دارید وگرنه کارت با پیغامی ضبط می‌شود. در این برنامه برای دریافت ۳ بار گذرواژه، از حلقه for استفاده شده است. هر بار که گذرواژه از ورودی دریافت می‌شود، با گذرواژه اصلی مقایسه می‌شود و در صورت صحیح بودن علامتی به نام متغیر flag مقدار true می‌گیرد. این متغیر در آغاز برنامه مقدار false دارد و تا زمانی که گذرواژه درست وارد نشود همچنان false می‌ماند.


```

using System;
class ATMGetPass
{
    static void Main ()
    {
        int i;
        string mainPassword = "1404", password;
        bool flag = false;
        for (i = 3; i >= 1; i--)
        {
            Console.WriteLine("Enter Password: ");
            password = Console.ReadLine();
            Console.WriteLine();
            if (password == mainpassword)
            {
                flag = true;
                Console.BackgroundColor = ConsoleColor.DarkGreen;
                Console.WriteLine(" That is right...");
                Console.Beep();
                break; // خروج از حلقه
            }
            else
                Console.WriteLine(" PLZ try again... ");
        }
        if (flag = false)
        {
            Console.BackgroundColor = ConsoleColor.Red;
            Console.WriteLine(" Your card will no longer work!!!");
        }
    }
}

```

سؤال: چرا ترجیح می‌دهیم در این برنامه از متغیر رشته‌ای حاوی رشته‌ای از ارقام، برای دریافت گذرواژه استفاده کنیم؟

سؤال: به‌طور کلی چه زمانی از متغیرهای عددی و چه موقع از متغیر رشته‌ای رقمی در برنامه کمک می‌گیریم؟

گسترش برنامه: متغیر گذرواژه اصلی را به جای عدد، با حروف مقداردهی نمایید و برنامه را طوری تغییر دهید که نسبت به دریافت حروف کوچک و بزرگ حساس نباشد. برای این کار، از متدهای `ToLower()` و `ToUpper()` در شرط استفاده کنید.

سؤال: عملکرد متدهای `ToLower()` و `ToUpper()` را توضیح دهید.

نکته

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!");
```

۲-۲-۷- کاربرد دستور `continue` در ساختار `for` :

مثال ۲-۹-۷: همه شما بازی هپ را می‌شناسید و بارها برای تمرین جدول ضرب در دوران ابتدایی این بازی را انجام داده‌اید. در این برنامه می‌خواهیم بازی هپ را برای عدد ۵ شبیه‌سازی کنیم. **الگوریتم** یا روش انجام کار: در بازی هپ، هر جا که به مضرب عدد تعیین شده می‌رسیم باید پیغام هپ را چاپ کنیم. برای این کار، حلقه‌ای را تا مثلاً ۵۰ در نظر می‌گیریم تا هر جا به مضرب ۵ رسید، هپ با رنگ دیگری چاپ شود. برای رسیدن به مضرب ۵ از شرطی با عملگر `%` برای باقیمانده کمک می‌گیریم. در نگاه اول برنامه با آنچه تاکنون آموخته‌اید قابل نوشتن است.

```

using System;
class hop
{
    static void Main (string [ ] args)
    {
        for (int i = 1; i<=20; i++)
        {
            if (i % 5 ==0)
            {
                Console.ForegroundColor = ConsoleColor.Yellow;
                Console.WriteLine("hop");
                Console.ForegroundColor = ConsoleColor.Gray;
            }
            Console.Write("{0} ", i);
        }
        Console.ReadKey();
    }
}

```

برنامه ۹-۷ بازی هپ

سؤال: اشکال برنامه فوق چیست؟

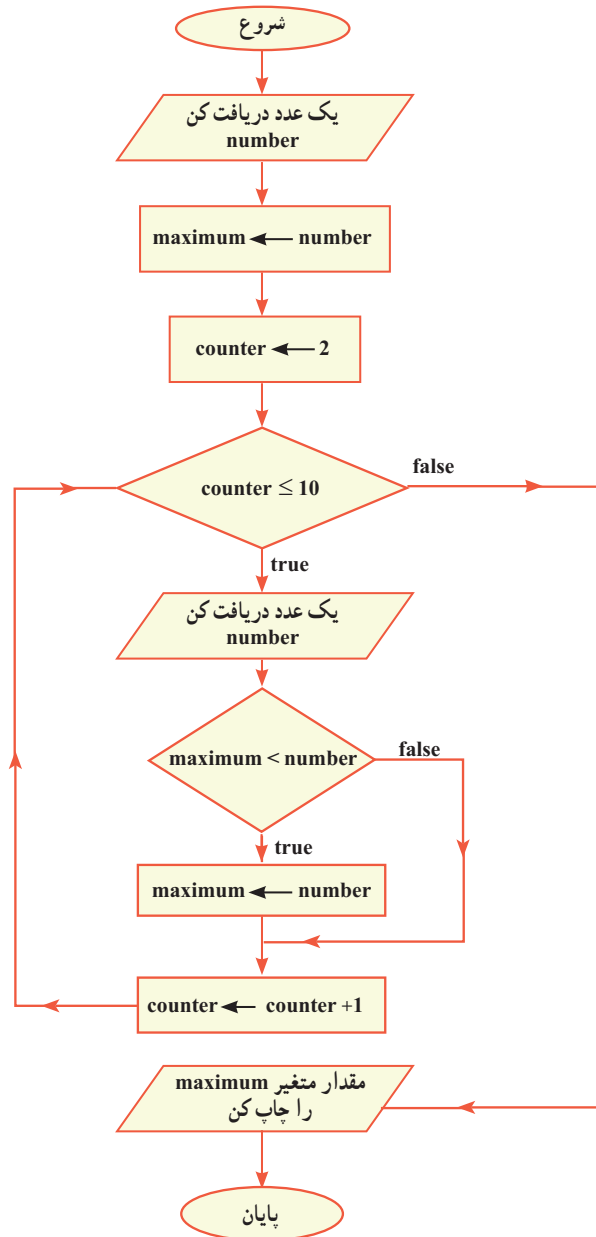
کلودو کارگاه ۳

- ۱- قطعه برنامه ۳-۷ را به یک برنامه کامل تبدیل کرده و آن را اجرا کنید.
- ۲- به برنامه ۵-۷، 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 می تواند تعداد دفعات را شمارش کند.
- ۳- قطعه برنامه های ۶-۷ و ۷-۷ را به یک برنامه کامل تبدیل کرده و آن را اجرا کنید.

در مثال ۹-۷ نباید هم مضرب عدد ۵ و هم هپ هر دو چاپ شوند. در واقع به جای مضرب باید هپ چاپ شود. بنابراین باید برنامه را در رسیدن به مضرب ۵ متوقف کنیم و از گام بعدی حلقه، تکرار را از سر بگیریم. در واقع زمانی از دستور پرش continue استفاده می‌کنیم که خواهان متوقف شدن یکی از مراحل تکرار و شروع بلافاصله مرحله بعدی تکرار هستیم. دستور continue مرحله فعلی را متوقف و مرحله بعدی تکرار را شروع می‌کند. در برنامه هپ کافیسست دستور continue را قبل از بسته شدن بلاک if قرار دهید.

مثال ۱۰-۷: 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 ۳-۶، توضیح داده شد، استفاده می‌کنیم. شکل ۷-۷، فلوجارت پیدا کردن بزرگترین عدد، از بین ده عدد را نشان می‌دهد. در ابتدای این فلوجارت، 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 ۷-۷ - فلوجارت پیدا کردن بزرگ‌ترین عدد

برنامه ۷-۱۰ بر اساس فلوجارت شکل ۷-۷ نوشته شده است.

```
class FindMaximum
```

```
{  
    static void Main(string[] args)  
    {  
        string input;  
        int number, maximum;  
        Console.WriteLine("Enter a number: ");  
        input = Console.ReadLine();  
        maximum = int.Parse(input); // Suppose first number is maximum.  
  
        for (int i=2; i<=10; i++)  
        {  
            Console.WriteLine("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 ۷-۱۰ پیدا کردن بزرگ‌ترین عدد از بین ده عدد

کود و نگاه ۲

- ۱- با تغییراتی در برنامه $1-7$ ، کوچکترین عدد را پیدا کنید. نام متغیر `minimum` را برای کوچک‌ترین عدد استفاده کنید.
- ۲- برنامه $1-7$ را طوری بنویسید که هم بزرگترین و هم کوچک‌ترین عدد را از بین ده عدد پیدا کند.
- ۳- به برنامه قبلی دستوری اضافه کنید تا فاصله بین بزرگ‌ترین عدد و کوچک‌ترین عدد را پیدا کند.

۳-۷- حلقه‌های متداخل

درون حلقه ممکن است دستور یا دستوراتی وجود داشته باشند. دستور یا دستورات داخل حلقه به تعداد تکرار حلقه، انجام می‌شود. بنابراین اگر حلقه‌ای مثلاً ۵ بار تکرار شود، دستورات داخل آن ۵ بار انجام می‌شوند.

دستور داخل حلقه ممکن است خودش، یک حلقه باشد. بنابراین حلقه‌هایی تودرتو خواهیم داشت که حلقه داخلی به تعداد دفعات تکرار حلقه بیرونی، تکرار می‌شود.

سؤال: مثال‌های زیر را ببینید و تعداد تکرار هر کدام را مشخص کنید.

```
for (int i = 1; i <= 10; i++)
```

```
Console.WriteLine(i);
```

```
for (int i = 1; i <= 5; i++)
```

```
{
```

```
    input = Console.ReadLine();
```

```
    num = int.Parse(input);
```

```
    sum = sum + num;
```

```
}
```

```
for (int i = 5; i >= 1; i--)
```

```
    for (int j = 1; j <= 3; j++)
```

```
for (int i = 1; i <= 2; i++)
{
    j = 4;
    do
    {
        Console.WriteLine("{1},{1}", i, j);
        j -= 1;
    } while (j >= 1);
}
```

سؤال: تفاوت میان تعداد تکرار و دستورات درون حلقه را در مثال‌های (۱) تا (۴) با همکلاسان خود به بحث بگذارید.

جدول Trace مثال شماره (۳) به عنوان نمونه آورده شده است. این جدول را تکمیل کنید.

I	J	خروجی
۵	۱	
	۲	
	۳	
۴	۱	
	۲	
	۳	

اکنون فرض کنید می‌خواهیم برای یک دونده در مسابقه دو شبیه سازی انجام دهیم. در این مسابقه هر دونده باید دور تا دور ۳ زمین مسابقه بدود که البته هر زمین را نیز باید ۵ بار به طور کامل، دور بزند.

سؤال: پیشنهاد می‌کنید کدام یک از حلقه‌های قبلی را با اندکی تغییر برای این مسابقه به کار ببریم؟

سؤال: در زندگی روزمره با چه مثال‌هایی از حلقه‌های تودرتو یا متداخل روبرو هستید؟

مثال ۱۱-۷- در این برنامه قصد داریم یک ساعت دیجیتال را با ساعت و دقیقه شبیه سازی کنیم.

الگوریتم یا روش انجام کار: ابتدا بخش نمایش دقیقه را کدنویسی می‌کنیم. برای نمایش دقیقه نیاز به حلقه‌ای داریم که ۶۰ بار کار کند (چرا؟). این بخش با حلقه بسیار ساده قابل کدنویسی است.

```
for (int min = 0; min < 60; min++)
    Console.WriteLine("{0}", min);
```

با اجرای کد بالا دقیقه‌های یک ساعت شبیه سازی می‌شود. در صورتی که بخواهیم ۱۲ ساعت را شبیه سازی کنیم کفایت دستورات بالا را ۱۲ بار تکرار کنیم. بنابراین دستورات را درون حلقه‌ای می‌نویسیم که ۱۲ بار تکرار می‌شود.

```
for (int hour = 1; hour <= 12; hour++)
    for (int min = 0; min < 60; min++)
        Console.WriteLine("{0}", min);
```

با تغییراتی در دستور WriteLine می‌توانیم ساعت را هم نمایش دهیم. بنابراین برنامه به شکل زیر خواهد بود:

```
using System;
class clock
{
    static void Main()
    {
        for (int hour = 1; hour <= 12; hour++)
            for (int min = 0; min < 60; min++)
                Console.WriteLine("{1}: {1}", hour, min);
    }
}
```

برنامه ۱۱-۷- نمایش ساعت دیجیتال با استفاده از حلقه‌های متداخل

سؤال؟ برنامه را طوری تغییر دهید که قبل از تغییر ساعت شمار، ایست موقتی با دریافت کلید از کاربر داشته باشد.

خوبه آزادی فصل هشتم

- ۱- درستی یا نادرستی هر عبارت را تعیین کنید.
 - (الف) در حلقهٔ while دستورات، حداقل یک بار تکرار می‌شوند.
 - (ب) عبارت منطقی حلقه do در جلوی آن نوشته می‌شود.
 - (ج) ما می‌توانیم حلقه for با تکرار ۵ را با while هم بازنویسی کنیم.
 - (د) برای خروج از حلقه دستور break به ما کمک می‌کند.
 - (ه) اگر در انتهای for علامت ; قرار دهیم خطا رخ می‌دهد.
- ۲- جاهای خالی را با عبارات مناسب تکمیل کنید.
 - (الف) معمولاً برای تکرار معین از حلقه کمک می‌گیریم.
 - (ب) برای این که حلقه for به صورت بی‌نهایت تکرار شود، به شکل استفاده می‌کنیم.
 - (ج) در حلقهٔ do عبارت منطقی در جلوی نوشته می‌شود.
 - (د) تعداد تکرار حلقه‌های متداخل از تکرار هر حلقه به دست می‌آید.
- ۳- حلقهٔ زیر را با استفاده از for بنویسید.

```
int i = 11;
do
{
    Console.WriteLine(i);
} while (i <= 99)
i = 11+;
```

- ۴- حلقهٔ سؤال قبل را trace کنید و در یک جمله بنویسید چه کاری انجام می‌دهد؟
- ۵- 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;
}
```

۶- خروجی این قطعه برنامه را به دست آورید.

```
for (int i = 1; i<=3; i++)
{
    for (int j = 1; j<=3; j++)
        if(i==j)
            Console.WriteLine(1);
        else
            Console.WriteLine(0);
    Console.WriteLine( );
}
```

تمرینات برنامه‌نویسی فصل هشتم

- ۱- برنامه‌ای بنویسید که اعداد زوج از ۳ تا ۲۱ را به صورت نزولی نشان دهد. برنامه را طوری تغییر دهید که خروجی برنامه روی عدد ۱۶ متوقف شود.
- ۲- برنامه‌ای بنویسید که یک عدد صحیح را دریافت کند و سپس اعداد فرد از ۱ تا آن عدد را چاپ نماید.
- ۳- 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.
- ۴- برنامه‌ای بنویسید که نمرات درس انگلیسی دانش‌آموزان 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
- ۵- 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.

۶- برنامه‌ای بنویسید که نمرات 10° دانش‌آموز را که هر کدام ۵ درس دارند دریافت و مجموع و معدل هر دانش‌آموز را محاسبه و چاپ نماید.

۷- برنامه‌ای بنویسید که جدول ضرب 10×10 را روی صفحه نمایش دهد.

واژگان و اصطلاحات انگلیسی فصل هشتم

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	Counter	
۲	Loop	
۳	Loop Body	
۴	ATM	
۵	Flag	

پیوست‌ها

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 به صورت 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:

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

۲- 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 :

۱- از طریق موتور جست‌وجوگری نظیر گوگل، برنامه *Net Framework* . را جست‌وجو کرده و وارد سایت شرکت مایکروسافت شوید یا به آدرس زیر مراجعه کنید :

<http://www.microsoft.com/net>

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

پیوست ۳

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

(الف) اشکال در ترجمه برنامه

- ۱- 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).
- ۲- 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 متوجه 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 جست‌وجو، عملیات زیر را انجام دهید :

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

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

کنید.

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

۳- در پنجره System Properties روی گزینه Environment Variables کلیک کنید.

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

۵- در پنجره‌ای که ظاهر می‌گردد مسیر برنامه‌های مورد جست‌وجو 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 :

۱- نام فایل را درست تایپ نکرده‌اید.

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

پیوست ۴: لیست کلمات کلیدی و رزرو شده

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	

