فصل دوم

آرایه

بخاطر دارید که متغیر مکانی از حافظه است که مانند یک ظرف مقداری را در خود نگه می‌دارد و با ورود مقدار جدید، مقدار قبلی آن از بین می‌رود. برخی مواقع در برنامه‌ها نیاز به نگه‌داری داده‌ها داریم و گاهی مقدار این داده‌ها خیلی زیادند و نمی‌توانیم ده‌ها و یا حتی صدها متغیر تعریف کنیم. بنابراین لیستی از داده‌های همنام تعریف می‌کنیم. در این فصل می‌خواهیم از مفهوم آرایه[1] برای تعریف چنین متغیرهایی استفاده کنیم و اهمیت و کاربرد آنها را با چند مثال مورد توجه قرار دهیم.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند :

۱ــ آرایه و کاربرد آن را توضیح دهد.

۲ــ متغیری از نوع آرایه تعریف کند و آن را در برنامه به کار بندد.

۳ــ روش مرتب سازی حبابی را شرح داده و برنامه مرتب سازی عناصر آرایه را بنویسد.

۴ــ با استفاده از دستورforeach به عناصر آرایه و رشته دسترسی پیدا کند.

۵ــ روش‌های جست‌وجو را توضیح داده و تفاوت آنها را بیان نماید.

۶ــ کلید جستجو را با استفاده از برنامه جستجوی خطی و دودویی پیدا کند.

۱ــ Array

۱ــ۲ــ تعریف آرایه

به مثال زیر توجه کنید.

مثال ۱ــ۲ : فرض کنید می‌خواهیم برنامه‌ای بنویسیم که در یک کلاس ۱۶ نفری محاسبه کند چند درصد نمره‌ها، بالاتر از میانگین کلاس در درس برنامه سازی۲ هستند.

الگوریتم یا روش انجام کار : برای نوشتن این برنامه، پس از دریافت نمرات کل کلاس لازم است ابتدا مجموع و میانگین نمرات را محاسبه کنیم. سپس هر نمره را با میانگین، مقایسه کرده و محاسبه کنیم چه تعدادی از نمرات بیشتر از میانگین کلاس بوده‌اند. برای شمردن این نمرات از شمارنده استفاده می‌کنیم و سپس درصد می‌گیریم.

برای نوشتن این برنامه، ناگزیریم از نمراتِ کلاس دوبار استفاده کنیم :

ــ بار اول در خواندن ورودی‌ها.

ــ بار دوم مقایسهٔ ورودی‌ها با میانگین محاسبه شده.

طبق روال کتاب برنامه سازی۱ متغیری را از نوع اعشاری تعریف می‌کنیم و در یک حلقه for، ۱۶ بار دریافت کرده و جمع و میانگین را محاسبه می‌کنیم.

سؤال: آیا می‌توانیم برای بار دوم از داده‌های ورودی استفاده کنیم و عمل مقایسه را انجام دهیم؟

این نیازِ ما به داده‌های ورودی سبب می‌شود به جای تعریف یک متغیر اعشاری برای نمره و ۱۶ بار تکرار دریافت داده، از ۱۶ متغیر اعشاری که یک لیست پشت سر هم را تشکیل می‌دهند استفاده کنیم، تا در صورت نیاز به دفعات لازم از آنها بهره ببریم.

در مبحث حلقه‌ها یاد گرفتیم که برای دریافت ۱۶ نمره به صورت زیر عمل کنیم :

```
string input;
float mark, total =0;
for (int i = 0; i <= 15; i++)
{
    Console. Write ("Enter a mark: ");
    input = Console. ReadLine ();
    mark = float. Parse (input);
    total = total + mark;
}
```

سؤال: متغیر mark در این روش پس از پایان حلقه، چه مقداری را در خود دارد؟

در روش بالا ما به همه ۱۶ داده‌ای که درون یک متغیر ریخته شده است دسترسی نداریم. چرا؟ بنابراین چگونه می‌توانیم برای بار دوم به داده‌ها دسترسی پیدا کنیم؟ پاسخِ این سؤال در تعریف متغیر قرار دارد. ما نیازمند خانه‌های بیشتری از حافظه هستیم تا داده‌ها را به صورت تک تک در آنها قرار دهیم. این خانه‌های پشتِ سرِهم و همنام آرایه نام دارند.

تعریف آرایه : مکان‌های متوالی در حافظه کامپیوتر که در آن داده‌هایی از یک نوع نگهداری می‌شوند آرایه نام دارند. مثلاً ۱۶ مکان متوالی را در نظر بگیرید که در هر یک از آنها نمره درسی دانش‌آموزان یک کلاس قرار دارد.

17.5	15	19	16.5	.	.	.	20	14	18

هر یک از مکان‌ها، یک عنصرِ[۱] آرایه نامیده می‌شود. برای تفکیک و دسترسی به‌این مکان‌ها، از یک عدد صحیح به نام اندیس[۲] استفاده می‌شود. (مانند پلاک خانه‌های یک کوچه) اولین عنصر آرایه با اندیس صفر مشخص می‌شود و عنصر بعد از آن شمارهٔ یک و به ترتیب جلو می‌رود تا آخرین عنصر که یک واحد کمتر از تعداد عناصر آرایه است. بنابراین آرایه‌ای با ۵ عنصر دارای اندیس‌های 5، 4، 3، 2، 1، 0 می‌باشد. تعداد عناصر آرایه را اندازه یا طول[۳] آرایه می‌نامند.

سؤال: در مثال بالا اندیس آخرین عنصر آرایه چه عددی است؟

۲ــ۲ــ نحوه ایجاد یا تعریف آرایه در زبان C#

به خاطر دارید که برای تعریف یک متغیر از نوع float به شکل زیر عمل می‌کردیم :

```
float mark;
```

برای تعریف یک آرایه از نوع float مشابه تعریف یک متغیر به شکل زیر عمل می‌شود :

```
float[] mark;
```

۱ــ Element

۲ــ Index

۳ــ Length Or Size

اما تعریف بالا هنوز کامل نیست و فقط مرحلهٔ اولِ تعریف است. در زبان سی شارپ برای تعریف یک آرایه باید دو مرحله‌ای عمل کنیم :

شکل کلی مرحله اول تعریف آرایه :

نام متغیر آرایه [] نوع داده ;

در تعریف متغیر گفته شد مکانی از حافظه برای نگهداری داده‌هاست اما با تعریف بالا هنوز مکانی از حافظه برای آرایه تخصیص نیافته است. با استفاده از عملگر new و مشخص کردن اندازه آرایه در مرحله دوم، تعریف کامل می‌شود.

شکل کلی مرحله دوم تعریف آرایه :

نام متغیر آرایه = new نوع داده [اندازه آرایه] ;

بنابراین برای تعریف متغیرmark با ۲۰ عنصر به صورت زیر عمل می‌کنیم :

```
float[] mark;          ← مرحله اول
new float [20]         ← مرحله دوم
```

به جای دو دستور بالا می‌توانیم دستور زیر را جایگزین نماییم :

```
float[ ] list = new float [20];
```

شکل کلی تعریف آرایه :

نام آرایه [ ] = new نوع داده [اندازه آرایه] ;

۳ــ۲ــ دسترسی، مقداردهی و نمایش عناصر آرایه

۱ــ۳ــ۲ــ دسترسی به عناصر آرایه : برای دسترسی به عناصر آرایه، از نام آرایه با

ذکراندیس به صورت زیر استفاده می‌کنیم :

نام متغیر آرایه [اندیس] ;

برای مثال list[0] بیانگر اولین عنصر آرایه و list[19] آخرین عنصر، که بیستمین عضوِ آرایه list است.

۲ــ۳ــ۲ــ مقداردهی عناصر آرایه : برای مقداردهی عناصر آرایه به چند روش می‌توان عمل کرد. یک روش مقداردهی، با استفاده از دستور انتساب است. برای مثال ذخیره کردن نمره 17.5 در اولین عنصر آرایه را با استفاده از دستور انتساب مشاهده می‌کنید.

```
list[0]=17.5f;
```

سؤال: به یاد دارید که چرا در انتهای عدد 17.5 حرف f قرار دارد؟

برای مقداردهی هر عنصر آرایه در طول اجرای برنامه، می‌توانید داده‌ای را از کاربر دریافت کرده و آن را در عنصری از آرایه نگهداری کنید. مثلاً نمره‌ای را از کاربر دریافت و در سومین عنصر ذخیره می‌کنیم :

```
string input;
float [] list = new float [20];
Console. Write ("Enter your mark: ");
input = Console. ReadLine();
list[2] = float. Parse (input);
```

عنصر سوم ←

و یا برای دریافت تمام نمرات و قراردادن آنها در آرایه list به صورت زیر عمل می‌کنیم :

```
for (int i = 0; i < 20; i++)
{
    Console. Write ("Enter a mark: ");
    input = Console. ReadLine ();
    list [i] = float. Parse (input);
}
```

روش دیگر مقداردهی عناصر آرایه در هنگام تعریف و ایجاد آرایه است که در انتهای تعریف آرایه در بین علامت‌های { } مقدار هر عنصر از آرایه را به ترتیب معین می‌کنیم :

نام آرایه [] نوع داده = new نوع داده [] {مقدار، ...، مقدار، مقدار} ;

همان‌طور که مشاهده می‌کنید عددی بین علامت‌های [ ] وجود ندارد، بنابراین اندازه آرایه را ذکر نمی‌کنیم بلکه اندازه چنین آرایه‌هایی با تعداد مقادیر نوشته شده بین علامت‌های { } تعیین می‌شود. مثلاً برای ایجاد آرایه‌ای برای سکه‌های رایج بر حسب ریال، به صورت زیر تعریف می‌کنیم.

```
int [] coin = new int [] {500,1000, 2000, 5000};
```

دستور بالا آرایه‌ای به نام coin شامل چهار عنصر ایجاد می‌کند که هر عنصر آن مقدار یک سکه را مشخص می‌کند که از سکه ۵۰۰ ریالی شروع و به سکه ۵۰۰۰ ریالی ختم می‌شود.
دستور بالا را می‌توان با دستورات زیر جایگزین نمود :

```
int[] coin = new int [4];

coin [0] = 500;
coin [1] = 1000;
coin [2] = 2000;
coin [3] = 5000;
```

همچنین با توجه به اینکه آرایه coin دارای مقادیر اولیه مشخص است می‌توان بدون استفاده از عملگر new آن را ایجاد کرد. بنابراین دستورات بالا را می‌توان با دستور زیر جایگزین نمود :

```
int[] coin = {500, 1000, 2000, 5000};
```

نکته

بعد از ایجاد آرایه، نمی‌توانید اندازه آن را تغییر دهید یعنی نمی‌توانید عنصری به آن اضافه و یا کم کنید.

بنابراین می‌توان در تعریف آرایه‌هایی که از قبل مقادیر اولیه‌شان مشخص است بدون استفاده از عملگر new به شکل زیر عمل کرد :

نام آرایه [] نوع داده = { مقدار، ...، مقدار، مقدار } ;

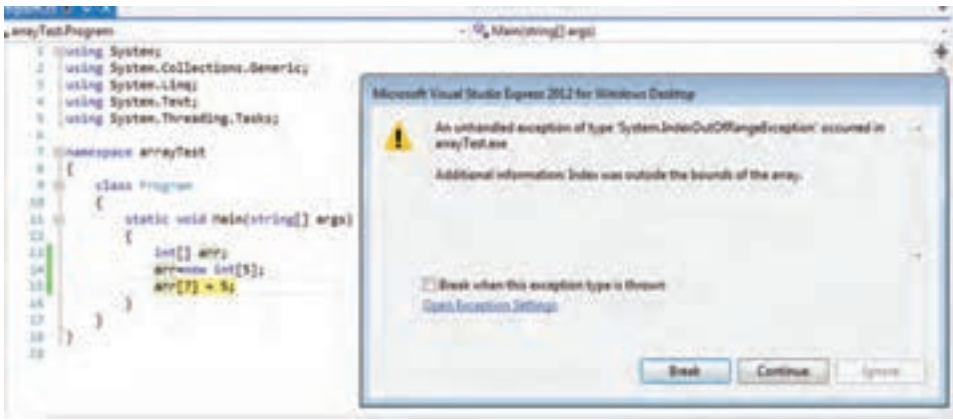
۳ــ۳ــ۲ــ نمایش عناصر آرایه : برای نمایش محتوای عناصر آرایه می‌توان از متد Write() یا WriteLine() استفاده نماییم. مثلاً برای نمایش محتوای آخرین عنصر آرایه نمرات، خواهیم داشت :

```
System. Console. WriteLine ("Last mark: " + list [19]);
```

سؤال: عناصر اندیس‌های زوج این آرایه را با حلقه for نمایش دهید؟

نکته

در زبان C# محدوده اندیس آرایه کنترل می‌شود و نباید از عدد صفر کمتر و همچنین از اندازه آرایه بیشتر یا مساوی باشد. اگر برنامه‌نویس اشتباه کند و اندیس بالاتری را استفاده کند در هنگام ترجمه برنامه با خطا روبه‌رو می‌شود (شکل ۱ــ۲).



شکل ۱ــ۲ــ خطای سرریزی محدوده اندیس آرایه

کار در کارگاه ۱

در آغاز فصل، مثال ۱ــ۲ را با الگوریتم ذکر کردیم. اکنون می‌خواهیم مرور دوباره داشته باشیم و برنامه را در محیط VS بنویسیم. می‌خواستیم برنامه‌ای بنویسیم که در یک کلاس ۱۶ نفری محاسبه کند چند درصد نمره‌ها، بالاتر از میانگین کلاس در درس برنامه‌سازی۲ هستند.

الگوریتم یا روش انجام کار : برای نوشتن این برنامه، پس از دریافت نمرات به وسیلهٔ حلقه در آرایه و محاسبه میانگین، در حلقهٔ دوم، عمل مقایسه صورت می‌گیرد. هر نمره با میانگین مقایسه شده و درصورتی که نمره بیشتر از میانگین کلاس باشد، به وسیلهٔ یک متغیر به عنوان شمارنده، تعداد را محاسبه می‌کنیم و سپس درصد می‌گیریم.

```
using System;

class Program
{
  static void Main(string[] args)
  {
    float[] mark;
    mark = new float[15];
    float avg, percen, sum = 0;
    byte counter = 0;

    for (int i = 0; i <= 15; i++)
    {
      Console.Write("enter the {0}th mark: ", i + 1);
      mark[i] = float.Parse(Console.ReadLine());
      sum = sum + mark[i];
    }
    avg = sum / 16;
    for (int i = 0; i <= 15; i++)
      if (mark[i] > avg)
        counter++;
    Console.WriteLine(" {0} person(s) > AVG ", counter);
    Console.WriteLine("-------------------");
```

```
            Console.WriteLine();
            percen = (counter / 16f) * 100
            Console.WriteLine("     {0} % > AVG ", percen);
            Console.ReadKey();

        }
    }
```

برنامه ۱ــ۲ــ محاسبه درصد نمرات بالاتر از میانگین کلاس

۴ــ۲ــ حلقه foreach

برای مواقعی که بخواهیم محتوای تمام عناصر آرایه را مورد استفاده قرار دهیم یا آنها را روی صفحه نمایش ببینیم، می‌توان به جای حلقه for ازحلقه foreach استفاده کنیم. دستور foreach روی هر عنصر از داده قابل شمارش تکرار می‌شود. مثلاً هر دو نوع array و string قابل شمارش هستند.

شکل کلی این دستور به صورت زیر است :

foreach (نوع داده نام متغیر in نام آرایه)

دستور ؛

دستور زیر عناصر آرایه score را روی صفحه نمایش می‌دهد.

```
foreach (int  aValue  in  score)
        Console.WriteLine(aValue) ;
```

همان‌طور که مشاهده می‌کنید در هنگام دسترسی به عناصر آرایه از اندیس کمک نمی‌گیریم بلکه با هر بار تکرار حلقه، متغیر aValue نقش یک عنصر از آرایه را به عهده می‌گیرد که با کلمه کلیدی in به آرایه score متصل می‌شود.

مثال ۲ــ۲ : برنامه‌ای بنویسید که نمرات دانش‌آموزان را دریافت کند و تعداد مردودی‌ها را نمایش دهد.

الگوریتم یا روش انجام کار : پس از تعریف آرایه و دریافت نمرات، توسط دستور foreach روی عناصر آرایه حرکت می‌کنیم و شرط مردودی که نمره کمتر از ۱۲ است را بررسی کرده و از متغیر شمارنده کمک می‌گیریم (برنامه ۲ــ۲).

```
using System;

  class MarkOfSudents
  {
    static void Main(string[] args)
    {
      byte counter = 0;
      float[] mark = new float[15];
      for (int i = 0; i < 15; i++)
      {
        Console.Write("Enter the mark of {0}th student :  ", i + 1);
        mark[i] = float.Parse(Console.ReadLine());
      }
      foreach (float m in mark)
        if (m < 12) counter++;
      Console.WriteLine(" - - - - - - - - - - - - - - - - - - - - - - ");
      Console.WriteLine(" the number of failed student(s) is : " + counter);
      Console.ReadKey();
    }
  }
```

برنامه ۲ــ۲ــ دریافت نمرات و شمارش مردودی‌ها

مثال ۳ــ۲ : رشته‌ای را از ورودی دریافت کنید و تک تک کاراکترهای آن را در خطوط جداگانه چاپ کنید.

الگوریتم یا روش انجام کار : با توجه به‌اینکه داده رشته‌ای نیز مانند آرایه از عناصر قابل شمارشی تشکیل شده است، می‌توان با استفاده از دستورforeach به تک تک عناصر آن دسترسی پیدا

کرد. بنابراین رشته‌ای را دریافت می‌کنیم و با استفاده از متغیر نوع کاراکتر به تک تک کاراکترهای رشته دست می‌یابیم (برنامه ۳ــ۲).

```
using System;

  class Testforeach
  {
    static void Main()
    {
      string name;
      Console.Write("Enter a Name:");
      name = Console.ReadLine();
      foreach (char ch in name)
        Console.WriteLine(ch);
    }
  }
```

برنامه ۳ــ۲ــ نمایش کاراکترهای رشته در خطوط جداگانه

ویژگی *Length*

آرایه names را در نظر بگیرید.

```
string[] names = new string[10];
```

طول آرایه names برابر ۱۰ است. در زبان C# می‌توان توسط ویژگی[۱] Length از اندازه یک آرایه یا تعداد عناصر آن مطلع شد.

نام آرایه · Length

مثال ۴ــ۲ : در قسمت آزمون چندگزینه‌ای یک برنامه آموزشی، پاسخ صحیح سؤالات در آرایه‌ای به نام answer قرار دارد، می‌خواهیم پاسخ صحیح سؤالات یا محتوای عناصر این آرایه را نمایش دهیم.

الگوریتم یا روش انجام کار : برای نمایش عناصر آرایه answer از حلقه for استفاده

۱ــ Property

می‌کنیم. شمارنده حلقه باید از صفر شروع و تا آخرین عنصر آرایه که یک واحد کمتر از تعداد عناصر آرایه است پیش رود. بنابراین به تعداد عناصر آرایه نیاز داریم که از ویژگی Length استفاده می‌کنیم :

```
for (int i=0; i<answer. Length ; i++)
    Console. Write (answer [i]);
```

قطعه برنامه ۴ــ۲ــ نمایش پاسخ‌های صحیح آرایه answer

سؤال: به نظر شما این ویژگی در چه برنامه‌هایی ممکن است مورد استفاده قرار بگیرد؟

۵ــ۲ــ تعیین اندازه آرایه در هنگام اجرای برنامه

در مثال‌هایی که تاکنون انجام دادیم اندازه یا تعداد داده‌ها از قبل مشخص بود و اندازه آرایه در متن برنامه مشخص شده بود، اما حالتی را در نظر بگیرید که تعداد داده‌هایی که قرار است کاربر وارد کند قبل از اجرای برنامه نامشخص باشد در این حالت چه باید کرد؟ مثلاً می‌خواهیم برنامه‌ای بنویسیم که میانگین نمرات دانش‌آموزان کلاس را محاسبه کند و برای هر کلاس، با تعداد دانش‌آموزان مختلف، قابل استفاده باشد.

روش اول : می‌توانیم یک آرایه با تعداد عناصر زیاد مثلاً ۵۰ عنصر را پیش‌بینی کنیم و داده‌ها را از کاربر دریافت و در آرایه قرار دهیم و پس از ورود آخرین نمره، کاربر یک عدد خاص مثلاً عدد ۱- را برای پایان دادن به ورودی، دریافت کند و در این صورت حلقهٔ دریافت نمرات پایان یابد. در این روش با وارد کردن هر نمره نیز یک شمارنده را افزایش می‌دهیم تا تعداد نمرات را در اختیار داشته باشیم و در این صورت، ویژگی Length دیگر در حلقه کاربرد ندارد. چرا؟ در این روش همواره تعدادی از عناصر آرایه خالی و بدون استفاده هستند.

روش دوم : ابتدا تعداد نمرات را از کاربر سؤال کنیم و سپس آرایه‌ای به همان میزان ایجاد کرده و با استفاده از حلقه for مانند مثال‌های قبلی اقدام به دریافت نمرات کرده و از ویژگی Length نیز به عنوان شرط انتهای حلقه استفاده کنیم.

مثال ۵ــ۲ : می‌خواهیم برنامه‌ای بنویسیم که میانگین نمرات دانش‌آموزان یک کلاس با هر تعداد دانش‌آموز را محاسبه کند.

الگوریتم یا روش انجام کار : برای نوشتن چنین برنامه‌ای از روش دوم که توضیح داده شد استفاده می‌کنیم. در برنامه زیر به نحوه تعریف آرایه list توجه کنید.

```
using System;

class StudentList
{
    static void Main()
    {
      string input;
      int listSize;
      Console. Write ("Enter number of the students: ");
      input = Console. ReadLine ();
      listSize = int.Parse (input);

      float [] list = new float [listsize];  ⇦
      float total = 0;

      for (int i = 0; i< list. Length ; i++)
      {
          Console. Write ("Enter a mark: ");
          input = Console. ReadLine();
          list[i] = float. Parse (input);
          total + = list [i];
      }
      Console. WriteLine ("Average of the marks: {0}", total / list.
Length);
    }
}
```

برنامه ۵ــ۲ــ محاسبه میانگین نمرات با تعداد نامشخص

۶ ــ ۲ــ مرتب کردن داده‌های یک لیست

الگوریتم‌های مختلفی برای مرتب کردن عناصر لیست وجود دارد بعضی از آنها خیلی ساده و ابتدایی هستند مانند روش مرتب‌سازی حبابی[۱] که زمان زیادی طول می‌کشد تا لیست مرتب شود و برخی از روش‌ها وجود دارند مانند روش مرتب‌سازی سریع[۲] که سرعت بسیار بالایی دارد.

مثال ۶ــ۲ : نمرات دانش‌آموزان یک کلاس را دریافت نموده و به صورت صعودی مرتب کنید. سپس رتبه‌های اول تا سوم را اعلام کنید.

الگوریتم یا روش انجام کار : فرض کنید می‌خواهیم لیستی از نمرات دانش‌آموزان را که به صورت زیر است مرتب کنیم :

۱۷, ۱۴, ۸/۵, ۱۳, ۲۰

بعد از مرتب کردن، لیست به صورت زیر خواهد شد :

۸/۵, ۱۳, ۱۴, ۱۷, ۲۰

در این روش از ابتدای لیست شروع کرده و دو عنصر اول و دوم را با یکدیگر مقایسه می‌کنیم. اگر ترتیب آنها درست نبود، آنها را جابه‌جا می‌کنیم. سپس به سراغ عنصر دوم و سوم می‌رویم و عمل مقایسه و در صورت لزوم جابه‌جایی را انجام می‌دهیم. و تا آخر لیست به همین ترتیب جلو می‌رویم. در الگوریتم مرتب سازی همواره دو عمل «مقایسه»[۳] و «جابه جایی»[۴] انجام می‌شود.

سؤال: وقتی که به انتهای لیست می‌رسیم چه اتفاقی می‌افتد؟

اکنون اعداد مورد نظر را به روش حبابی و به‌ترتیب صعودی یعنی از کوچک به بزرگ مرتب می‌کنیم :

۱ــ Bubble Sort Method

۲ــ Quick Sort Method

۳ــ Compare

۴ــ Swap

شکل ۲ــ۲ــ مراحل مرتب سازی حبابی، فاز اول

توجه کنید که پس از انجام این تعداد مقایسه و جابه‌جایی، بزرگ‌ترین عددی که در لیست وجود دارد خود را به جای اصلی خود، یعنی در انتهای لیست می‌رساند.

سؤال: چرا به‌این روشِ مرتب‌سازی، حبابی گفته می‌شود؟

هنوز لیست به طور کامل مرتب نشده و باید مجدداً از ابتدای لیست شروع کرده و تا انتهای لیست عمل مقایسه و جا به‌جایی را انجام دهیم تا عدد بزرگ‌تر بعدی نیز، خود را به مکان صحیح برساند. در فاز دوم، عددِ بزرگِ دوم، به مکانِ یکی مانده به انتهای لیست منتقل می‌شود.

بنابراین باید، عملیات مقایسه و جابه‌جایی را بارها تکرار کرد تا در نهایت، لیست به طور کامل مرتب شود.

سؤال: در مثال ذکرشده چند فاز باید طی شود تا اعداد در جای خود قرار گیرند؟ مراحل بعدی را مانند نمونه بنویسید.

توجه داشته باشید که عمل مرتب سازی یک عمل وقت‌گیر است به‌خصوص روش حبابی که یک روش کند است، چون تعداد عمل مقایسه و جا به جایی در آن بسیار زیاد است.

سؤال: در مرتب‌سازی حبابی، وقت‌گیرترین و بدترین حالت آرایشِ اعدادِ ورودی کدام است؟

اکنون سعی می‌کنیم الگوریتم را پیاده‌سازی کنیم. همان‌طور که در قطعه برنامه زیر مشاهده می‌کنید در فاز اول ۴ مرحله مقایسه و جابه جایی داریم. چرا؟ برای این که مراحل بالا را پیاده سازی کنیم به حلقه‌ای نیاز داریم که در هر بار اجرای حلقه، عدد زم را با عدد ۱+زم مقایسه کند و در صورتی که بزرگ‌تر بود، جابه‌جایی انجام دهد. دستورات جابه‌جایی با استفاده از متغیر کمکیِ سوم را در سال گذشته تمرین کرده‌اید که یکی از کاربردهای آن در مرتب سازی است. اکنون دستور مربوط به‌این بخش را می‌نویسیم :

```
for (int j = 0;      ; j++)
    if (arr[j] > arr[j + 1])
    {
        temp = arr [j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
    }
```

قسمت شرط پایانی حلقه را فعلاً ننوشته‌ایم تا به توافق برسیم. اگر در پاسخ به سؤال بالا، مراحل مقایسه و جابه جایی را در فاز دوم و سوم و ... نوشته باشید، متوجه شده‌اید که حلقهٔ بالایی باید به تعداد فازهای مرتب‌سازی تکرار شود. بنابراین باید حلقهٔ بالا را درون حلقهٔ دیگری قرار دهیم که مشخص‌کنندهٔ فازهای تکرار است.

سؤال: چرا تعداد تکرار حلقهٔ بیرونی که معرف فازهای مرتب سازی است، یکی کمتر از تعدادِ اعداد است؟

```
for (int i = 4; i > 0 ; i--)
    for (int j = 0;      ; j ++)
        if (arr [j] > arr[j + 1])
        {
            temp = arr [j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
        }
```

اکنون نوشتن حلقه‌های مرتب‌سازی حبابی تمام شده است اما هنوز نمی‌دانیم حلقهٔ داخلی که نشان دهندهٔ مراحل در هر فاز است، چه شرطی برای پایان دارد. اگر به واژهٔ حبابی توجه کنیم در می‌یابیم که در پایان هر فاز، یک عنصر در مکان صحیح خود در لیست جای می‌گیرد. پس دیگر نیازی به مقایسهٔ این عنصر نداریم زیرا به درستی در مکان خود قرار گرفته و مرتب شده است. بنابراین در فاز بعدی یک مرحله کمتر از فاز قبل، مقایسه و جابه جایی داریم که با شرط j<i می‌نویسیم. برنامه ۶ــ۲ را ببینید و با هم کلاس خود برای اعداد قبلی Trace کنید. (برنامه ۶ــ۲)

سؤال: دستوراتی به برنامه اضافه نمایید تا رتبه‌های اول تا سوم را چاپ نماید.

```
using System;

class BubbleSort
{
   static void Main(string[] args)
   {
      Console.Write("enter the array size: ");
      int arrsize = int.Parse(Console.ReadLine());

      int[] arr;
      arr = new int[arrsize];

      for (int i = 0; i < arrsize; i++)
      {
         Console.Write("enter the {0}th number: ", i+1);
         arr[i] = int.Parse(Console.ReadLine());
      }

      //Bubble Sort.....................................

      int temp = 0;
      for (int i = arrsize - 1; i > 0; i--)
      {
         for (int j = 0; j < i; j++)
            if (arr[j] > arr[j + 1])
            {
               temp = arr[j];
               arr[j] = arr[j + 1];
               arr[j + 1] = temp;
            }
      }

      // show  the sorted array
      Console.WriteLine();
      Console.WriteLine("  the sorted array is:    ");
      Console.WriteLine("-------------------------");
      for (int i = 0; i < arrsize; i++)
         Console.WriteLine("arr[{0}] = {1}", i+1, arr[i]);
   }
}
```

مقایسه

و

جابجایی

برنامه ۶ــ۲ــ مرتب‌سازی حبابی

۷ــ ۲ــ عمل جستجو در لیست

جستجو پرکاربردترین عملیاتی است که روی لیست‌ها انجام می‌شود و به منظور یافتن عنصری در آرایه صورت می‌گیرد که به آن کلید جستجو گفته می‌شود. در این کتاب ما به توضیح و برنامه نویسی دو نوع جستجو می‌پردازیم.

۱ــ۷ــ۲ــ جستجوی خطی(ترتیبی)[۱] :

ساده‌ترین روش جستجو، به‌صورت ترتیبی می‌باشد که در آرایه‌های کوچک و یا مرتب نشده مورد استفاده قرار می‌گیرد. در این روش جستجو، هر عنصر از آرایه با کلید جستجو مقایسه می‌شود. و برای تعیین اینکه کلید مورد نظر در آرایه وجود دارد یا نه، برنامه باید کلید را با تمام عناصر آرایه مقایسه کند. بنابراین در صورتی که کلید یافت شود نیازی به ادامهٔ جستجو نیست و می‌توان از برنامه خارج شد.

مثال ۷ــ۲ : برنامه‌ای بنویسید که نمره ۲۰ را از میان لیست نمرات کلاس پیدا کرده و اعلام کند مربوط به نفر چندم لیست است؟

الگوریتم یا روش انجام کار : ابتدا آرایه‌ای از نمرات دانش‌آموزان تعریف و به‌وسیلهٔ حلقه for نمرات را دریافت می‌کنیم. سپس برای جستجوی ترتیبی، با استفاده از حلقهٔ for تک تک عناصر را با کلید جستجو مقایسه کرده و در صورت یافتن کلید به کار حلقه پایان می‌دهیم. برای یافتن کلید از متغیر بولی found کمک می‌گیریم. این متغیر مقدار اولیه false دارد تا زمانی که کلید پیدا شود، در صورتی که کلید یافت نشود، هم چنان مقدار false خواهد داشت.[۲]

۱ــ Liner (Sequential) Search

۲ــ به چنین متغیرهایی که وضعیت دوحالتی خاموش یا روشن را نشان می‌دهند Flag گفته می‌شود. استفاده از تکنیک Flag در برنامه‌های زیادی کاربرد دارد.

```
using System;
  class Search
  {
    static void Main(string[] args)
    {
      Console.Write("enter the array size: ");
      int arrsize = int.Parse(Console.ReadLine());
      Console.WriteLine("-----------------");
      int[] arr;
      arr = new int[arrsize];
      //.....Initializing array.....
      for (int i = 0; i < arrsize; i++)
      {
        Console.Write("enter the {0}th number: ", i+1);
        arr[i] = int.Parse(Console.ReadLine());
      }
      //.....Linear Search.....
      bool found = false;
      int pos=-1;
      Console.WriteLine("-----------------");
      Console.Write("enter the search key: ");
      int key = int.Parse(Console.ReadLine());
      Console.WriteLine("= = = = = = = = =");
      for (int i = 0; i < arrsize; i++)
        if (arr[i] == key)
        {
          found = true;
          pos = i+1;
          break;
        }
      if (found)
        Console.WriteLine("the key found in  position number {0}  ", pos);
      else
        Console.WriteLine("the key not found");
      Console.ReadKey();
    }
  }
```

۲ــ۷ــ۲ــ جستجوی دودویی(باینری)[۱] :

روش دیگر جستجوی کلید در بین عناصر آرایه، روش جستجوی دودویی است. فرض کنید می‌خواهید شماره پلاک یک خانه را در یک کوچه بیابید، آنچه باعث می‌شود سریع‌تر خانهٔ مورد نظر را پیدا کنید، مرتب بودن شماره پلاک‌ها در کوچهٔ مورد نظر است.

سؤال: اگر در یک کوچهٔ طولانی دنبال یک خانه بودید و این شماره‌ها مرتب نبود، چه می‌کردید؟

روش جستجوی دودویی، روی آرایه‌های مرتب قابل اجراست. برای استفاده از روش جستجوی دودویی، در صورتی که آرایه مورد نظر نامرتب باشد، ابتدا باید آن را مرتب کنید. در مثال‌های قبل با روش مرتب سازی حبابی آشنا شدید. یکی از کاربردهای مرتب سازی در جستجوی دودویی است. این روش جستجو برای آرایه‌های بزرگ مناسب است. چرا؟

سؤال: به طور کلی چرا از این نوع جستجو استفاده می‌کنیم؟

مثال ۸ــ۲ : برنامه‌ای بنویسید که با دریافت کلید جستجو به روش دودویی آن را جستجو کنید.

الگوریتم یا روش انجام کار : اگر بازی حدس عدد را در کتاب برنامه سازی۱ به خاطر بیاورید، برای حدسِ عددی که بازیکن اول در ذهن سپرده بود، از روش حرکت کردن در لیست اعداد استفاده می‌کردیم. در ذهن فرد حدس زننده این بود که اعداد مرتب هستند. بنابراین به محض حدس اولین عدد منتظر نتیجه از بازیکن اول می‌ماند تا بگوید عدد حدس زده شده از کلید جستجو بزرگ‌تر است یا کوچک‌تر.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8.5	9	10	11	12.75	14	14.5	15	16	16.5	17	17.5	18	19.5	19.75	20

مقایسه داده مورد نظر با عنصر وسط

شکل ۳ ــ۲ــ لیست اعداد مرتب به همراه شماره عنصر

به‌این ترتیب در بازی حدس عدد از تکنیک مشابه جستجوی دودویی استفاده می‌شود. در این

۱ــ Binary

جستجو برای یافتن کلید، ابتدا عنصر وسط لیست با کلید مقایسه می‌شود. این مقایسه ۳ نتیجه در بردارد :

۱ــ کلید با عنصر وسط لیست یکسان است. بنابراین کلید پیدا شده و در این‌صورت شماره یا اندیس عنصر وسط آرایه را یادداشت می‌کنیم.

۲ــ عنصر وسط لیست، کوچک‌تر از کلید است، بنابراین نتیجه می‌گیریم که داده در نیمه سمت چپ نمی‌باشد. در نتیجه باید منطقه مورد جستجو را به نیمه سمت راست آرایه محدود کنیم.

۳ــ عنصر وسط لیست، بزرگ‌تر از کلید است، بنابراین نتیجه می‌گیریم که داده در نیمه سمت راست لیست نمی‌باشد. در نتیجه باید منطقه مورد جستجو را به نیمه سمت چپ آرایه محدود کنیم.

پس از بررسی مشخص کردن منطقه جدید، مجدداً عمل جستجو را تکرار می‌کنیم یعنی عنصر وسط لیست جدید را مورد بررسی قرار می‌دهیم و یکی از ۳ حالت بالا رخ خواهد داد که بر آن اساس یا داده یافت می‌شود و یا مجدداً عمل جستجو در لیست جدیدی که نصفِ اندازهٔ قبلی است، تکرار می‌شود.

در روش جستجوی دودویی با یک عمل مقایسه، منطقه مورد جستجو به‌اندازه نصف، کوچک می‌شود. بنابراین با چند عمل مقایسه یک لیست بزرگ از داده‌ها به منطقه کوچکی ختم می‌شود بنابراین سرعت عملیات جستجو در این روش بسیار بالا است.

برای مثال اگر لیستی دارای ۱۰ هزار عنصر باشد در اولین مرحله ۵۰۰۰ عنصر کنار می‌روند. در مرحله دوم ۲۵۰۰ عنصر و در مرحله سوم ۱۲۵۰ عنصر کنار می‌روند. بنابراین خواهیم داشت :

مرحله جستجو	تعداد عناصر لیست	مرحله جستجو	تعداد عناصر لیست
۱	۱۰۰۰۰	۲	۵۰۰۰
۳	۲۵۰۰	۴	۱۲۵۰
۵	۶۲۵	۶	۳۱۳
۷	۱۵۷	۸	۷۹
۹	۴۰	۱۰	۲۰
۱۱	۱۰	۱۲	۵
۱۳	۳		

در این جستجو، با حداکثر ۱۳ بار مقایسه، کلید را پیدا می‌کنیم و یا مطمئن می‌شویم که در لیست، چنین داده‌ای وجود ندارد. جستجوی دودویی در مقایسه با روش جستجوی ترتیبی دارای سرعت فوق العاده بالایی است. چرا؟

سؤال: کلید جستجو، در لیست ۱۰۰۰۰عنصری مثال قبل در روش ترتیبی در چند مرحله جستجو می‌شود؟

نکته

توجه داشته باشید از روش جستجوی دودویی تنها زمانی می‌توانید استفاده کنید که عناصر لیست مرتب باشند. بنابراین اگر لیست مرتب نباشد و بخواهید در لیست بزرگی جستجو کنید، بهتر است ابتدا لیست را مرتب کنید و سپس به روش دودویی جستجو نمایید.

برای اینکه بتوانیم هر بار لیست را نصف کنیم، اندیس‌های آرایه را به سمت منطقه مورد نظر هدایت می‌کنیم. برای این منظور باید وسط لیست را پیدا کنیم و عملیات جابه‌جایی اندیس داشته باشیم.

```
middle = (first + last) / 2;   محاسبهٔ اندیس وسط لیست
last = middle −1;   عملیات جابه جایی اندیس زمانی که منطقه جستجو سمت چپ باشد
first = middle +1;   عملیات جابه جایی اندیس، زمانی که منطقه جستجو سمت راست باشد
```

برنامه جستجو را مشاهده کنید. این برنامه برای داده‌های مرتب شده کفایت می‌کند، اما همیشه برای احتیاط بهتر است قبل از جستجوی دودویی داده‌های وارد شده را مرتب کنید. چرا؟

```
using System;
  class BinSearch
  {
    static void Main(string[] args)
    {
      Console.Write("enter the array size: ");
      int arrsize = int.Parse(Console.ReadLine());
      Console.WriteLine("-----------------");
      int[] arr;
      arr = new int[arrsize];
      //.....Initializing arr ay.....
      for (int i = 0; i < arrsize; i++)
      {
        Console.Write("enter the {0}th number: ", i+1);
        arr[i] = int.Parse(Console.ReadLine());
      }
      //.....Binary Search.....
      int last = arrsize - 1;
      int first = 0;
      int middle = (first + last) / 2;
      bool found = false;
      Console.WriteLine("-----------------");
      Console.Write("enter the search key: ");
      int searchkey = int.Parse(Console.ReadLine());
      Console.WriteLine("= = = = = = = = =");
      while (last >= first)
      {
        middle = (first + last) / 2;
        if (arr[middle] == searchkey)
        {
          found = true;
          break;
        }
        else if (arr[middle] > searchkey)
          last = middle - 1;
        else
          first = middle + 1;
      }
      if (found)
        Console.WriteLine("the key found in  position number= {0} ", middle);
else
        Console.WriteLine("the search key not found");
      Console.ReadKey();
    }
    }
```

برنامه ۸ــ۲ــ درصورتی قابل استفاده است که داده‌ها، مرتب شده وارد شوند. در ورود داده‌ها دقت کنید.

خودآزمایی فصل دوم

الف) درستی یا نادرستی عبارات زیر را تعیین کنید.

۱ــ جستجوی دودویی فقط بر روی لیست‌های مرتب امکان پذیر است.

۲ــ جستجوی خطی برای آرایه‌های بزرگ مناسب است.

۳ــ دستور foreach فقط بر روی آرایه‌ها قابل استفاده است.

۴ــ اندیس اولین عنصر آرایه، ۱ است.

۵ــ از آرایه زمانی استفاده می‌کنیم که مجبوریم چندین بار از داده‌های ورودی در برنامه استفاده کنیم.

ب) جاهای خالی را با عبارات مناسب پرکنید.

۶ــ در عمل جستجو در آرایه به مقداری که جستجو می‌شود..... می‌گویند.

۷ــ در هر الگوریتم مرتب‌سازی همیشه دو عمل..... و ..... انجام می‌شود.

۸ــ تعداد عناصر آرایه را ..... آرایه می‌نامند.

۹ــ به هر مکان آرایه، یک ..... آرایه می‌گویند.

ج) به سوالات زیر پاسخ دهید.

۱۰ــ آرایه‌ای برای نام روزهای هفته تعریف کنید.

۱۱ــ آرایه‌ای برای نام ماه‌های سال تعریف کنید.

۱۲ــ برنامه زیر را Trace کنید و در یک خط بنویسید چه عملی انجام می‌دهد.

```
class Program
{
static void Main(string[] args)
{
int[] b = new int[5];
int[] a = new int[5];
int[] c = new int[5];
for (int i = 0; i <= 4; i++)
```

```
{
    a[i] = int.Parse(Console.ReadLine());
    b[i] = int.Parse(Console.ReadLine());
    if (a[i] > b[i])
        c[i] = a[i];
else
        c[i] = b[i];
    Console.WriteLine(c[i]);
}
  Console.ReadKey();
{
```

۱۳ــ آرایه زیر را در نظر بگیرید و معادل دستورات زیر را با روش دیگری از تعریف آرایه بازنویسی کنید.

```
string[] week = new string[7];
week[0] = "Saturday";
week[1] = "Sunday";
week[2] = "Monday";
week[3] = "Tuesday";
week[4] = "Wednesday";
week[5] = "Thursday";
week[6] = "Friday";
```

۱۴ــ دستوری بنویسید که تعداد عناصر آرایه week را نمایش دهد.

۱۵ــ برنامه زیر را Trace کنید و خروجی آن را بنویسید.

```
namespace arrayTest
{
  class Program
  {
```

```
        static void Main(string[] args)
        {

            int[] arr = { 10, 13, 15, 6, 8, 20 };
            arr[1] = 3;
            for (int i = 0; i < 6; i++)
            {
                arr[i] += 2;
                Console.WriteLine(arr[i]);
            }
            Console.ReadKey();

        }
    }
}
```

۱۶ــ با توجه به اعلان آرایه arr به سؤالات زیر پاسخ دهید.

```
int[] arr = { 10, 13, 15, 6, 8, 20 };
```

دستوری بنویسید که مقدار سومین عنصر آرایه را صفر کند.

آرایه arr چقدر حافظه اشغال می‌کند؟

نتیجه اجرای قطعه کد زیر چیست؟

```
int temp = arr[0];
arr[0] = arr[5];
arr[5] = temp;
```

۱۷ــ کدام یک از روش‌های جستجو سریع‌تر است؟ چرا؟

۱۸ــ در یک آرایه مرتب با ۱۰ عنصر حداکثر با چند مقایسه نتیجه جستجو مشخص می‌شود؟

تمرینات برنامه‌نویسی فصل دوم

۱ــ با استفاده از حلقه foreach نام روزهای هفته را چاپ کنید.

۲ــ برنامه‌ای بنویسید که نمرات درس برنامه‌سازی یک کلاس ۱۵ نفره را دریافت کند و بیشترین و کمترین نمره کلاس را پیدا کرده و نمایش دهد.

۳ــ برنامه قبل را توسعه دهید تا علاوه بر بیشترین و کمترین نمره کلاس، اعلام کند چندمین دانش‌آموز بیشترین یا کمترین نمره را دارد.

۴ــ شرکت مخابرات می‌خواهد به مشتریانی که در طول یک سال، حداقل ۶ ماه، قبض تلفن همراه خود را قبل از بیستم ماه پرداخت می‌کنند به عنوان جایزه یک بسته مکالمه رایگان ۶۰۰ دقیقه‌ای هدیه دهد. برنامه‌ای بنویسد که برای یک مشتری روز پرداخت قبض در هر ماه را دریافت و اعلام کند که بسته مکالمه رایگان به مشتری تعلق می‌گیرد یا نه؟

۵ــ صاحب یک کارگاه آهنگری می‌خواهد به کارگرانی که بیش از ۳ فرزند دارند، مبلغ ۳۰۰۰۰۰ تومان مساعده پرداخت کند. برنامه‌ای بنویسید که تعداد فرزندان ۱۸ کارگر این کارگاه را دریافت کرده و تعداد کارگرانی که بیش از ۳ فرزند دارند را محاسبه کند و نمایش دهد. سپس کل مبلغ مساعده را نمایش دهد.

۶ــ برنامه قبل را توسعه دهید تا برای هر کارگاه با هر تعداد کارگر و هر مبلغ مساعده قابل استفاده باشد.

۷ــ برنامه سؤال ۵ را با foreach بازنویسی کنید.

۸ــ برنامه ۲ را توسعه دهید تا نام و نمره درس برنامه‌سازی یک کلاس ۱۵ نفره را دریافت کند. سپس نام دانش‌آموزانی که کمترین و بیشترین نمره را دارند به همراه نمره آنها نمایش دهد.

۹ــ یک شرکت خصوصی ۳درصد حقوق کارمندان خود را به عنوان مالیات از آنها کسر می‌کند. برنامه‌ای بنویسید که میزان حقوق ۳۵ کارمند این شرکت را دریافت کند. سپس میزان خالص دریافتی هر کارمند را بعد از کسر مالیات محاسبه و مجدداً در آرایه ذخیره کند.

فعالیت

متن زیر از MSDN با موضوع آرایه برداشت شده است. آن را با کمک هم کلاسی خود ترجمه کنید و به کلاس ارائه نمایید.

Arrays in General

C# arrays are zero indexed; that is, the array indexes start at zero. Arrays in C# work similarly to how arrays work in most other popular languages There are, however, a few differences that you should be aware of.

When declaring an array, the square brackets ([]) must come after the type, not the identifier. Placing the brackets after the identifier is not legal syntax in C#.

```
int[] table; // not int table[];
```

Declaring Arrays

C# supports single-dimensional arrays, multidimensional arrays (rectangular arrays) The following example shows how to declare single-dimensional array:

Single-dimensional arrays:

```
int[] numbers;
```

Declaring them (as shown above) does not actually create the arrays. In C#, arrays are objects and must be instantiated. The following examples show how to create arrays:

Single – dimensional arrays:

```
int[] numbers = new int[5];
```

واژگان و اصطلاحات انگلیسی فصل دوم

ردیف	واژه انگلیسی	معنی به فارسی
۱	Array	
۲	Binary	
۳	Bubble Sort Method	
۴	Compare	
۵	Element	
۶	Index	
۷	Length	
۸	Liner Search	
۹	Sequeutial Search	
۱۰	Property	
۱۱	Quick Sort Method	
۱۲	Size	
۱۳	Swap	

فصل سوم

داده شمارشی، کلاس و متد

در این فصل علاوه بر آشنایی با نوع داده شمارشی، با مفاهیم متد و کلاس که از مفاهیم پایه برنامه‌سازی شئ‌گرا هستند به صورت مقدماتی آشنا خواهید شد. هدف از این فصل، نوشتن متد یا ایجاد یک کلاس نیست. شناخت اجزای یک کلاس و متد کمک می‌کند در کاربرد آنها بیشتر به تعاریف شان دقت کنید و آنها را درست به کار گیرید. بدیهی است ایجاد کلاس در کتاب برنامه سازی۳ مورد توجه خاص قرار می‌گیرد.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند :

۱ــ نوع داده شمارشی را توضیح دهد.
۲ــ از داده شمارشی در برنامه‌های خود استفاده کند.
۳ــ متد را تعریف کند و کاربرد آن را بیان نماید.
۴ــ از برخی متدهای آماده در برنامه‌های خود استفاده کند.
۵ــ کلاس، شئ و کاربرد آنها را توضیح دهد.
۶ــ شکل کلی یک کلاس و اجزای یک کلاس را نام برده و هر یک را توضیح دهد.
۷ــ کلاس‌های آماده سی‌شارپ را در برنامه‌های خود به کار بندد.

۱ــ۳ــ نوع داده شمارشی[۱]

تاکنون با داده‌های ساده کار کرده‌اید. داده شمارشی از داده‌های مرکب است که خود شامل یک سری داده است. در شکل ۱ــ۳ قسمتی از دو برنامه نشان داده شده است که در هر یک از آنها دستور switch استفاده شده است. اگر فرض کنیم هر کدام از برنامه‌ها صحیح کار می‌کنند و کارکرد یکسانی دارند شما کدام‌یک را ترجیح می‌دهید؟

```
switch (player)
{
    case 1:
        ⋮
    case 2:
        ⋮
    case 1:
        ⋮
    case 1:
        ⋮
}
```
قطعه برنامه الف

```
switch (player)
{
    case playerRank. Leader:
        ⋮
    case PlayerRank. Co–Leade:
        ⋮
    case PlayerRank. Elder:
        ⋮
    case PlayerRank. Member:
        ⋮
}
```
قطعه برنامه ب

شکل ۱ــ۳ ــ ساختار switch

همان طور که در شکل ۱ــ۳ مشاهده می‌کنید با نگاه به قطعه برنامه (الف) متوجه می‌شویم که مقدار متغیر player با اعداد ۱، ۲، ۳ و۴ مقایسه می‌شود. اما این سؤال پیش می‌آید که این اعداد، مربوط به چه موضوعی هستند؟ آیا شماره بازیکن است ؟ آیا فرصت بازی بازیکن است؟ و . . .

اما وقتی به قطعه برنامه (ب) نگاه می‌کنیم برداشت دقیق‌تری نسبت به برنامه داریم. مقدار متغیر

۱ــ Enumerated Type

player با مقادیر Leader ، Co_Leader ، Elder ، Member که چهار سطح و مقام در بازی[1] است، مقایسه می‌شود.

خوانایی برنامه (ب) بالاتر از برنامه (الف) است، به‌دلیل اینکه به جای اعداد ثابت و بی معنی، از کلمات و نام‌های با معنی استفاده شده است. در فصل اول کتاب با مفهوم و کاربرد ثابت‌ها در برنامه آشنا شدید و مشاهده کردید که چگونه خوانایی برنامه را بالا می‌برند. در این بخش نیز با نوع داده شمارشی آشنا می‌شوید که سبب بالابردن خوانایی برنامه می‌شوند.

نوع داده شمارشی مجموعه‌ای از چند نام دلخواه می‌باشد که حالت‌ها و مقادیر مختلف یک موضوع را نشان می‌دهد. مثلاً برای روزهای هفته به جای اعداد ۱ تا ۷ (یا ۰ تا ۶) از نام‌ها و کلمات معنی دار استفاده می‌کنیم. علاوه بر روزهای هفته، برای نام ماه‌های سال، مقام یا درجه یک بازیکن و مدرک تحصیلی اشخاص نیز می‌توان از نوع داده شمارشی استفاده کرد.

برای تعریف یک نوع داده شمارشی از کلمه کلیدی enum به صورت زیر استفاده می‌شود. نوع دسترسی معمولاً public است و روش نوشتن نام نوع داده نیز مطابق با روش پاسکال است.

محل قرارگیری تعریف نوع داده شمارشی، معمولاً خارج از کلاس و در ابتدای برنامه است.

```
نام دلخواه enum نوع دسترسی
{
        لیستی از نام‌ها و کلمات
}
```

نکته

در لیست نام‌ها و کلمات در نوع داده شمارشی، هر نام با علامت کاما از نام دیگر جدا می‌شود. نقطه ویرگول در این تعریف استفاده نمی‌شود.

در تعریف صفحه بعد نوع داده شمارشی برای نام روزهای هفته میلادی با دسترسی وسیع را مشاهده می‌کنید.

۱ــ بازی CoC

```
public enum DayOfWeek
{
    Sunday,
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday
}
```

هر یک از اعضای نوع داده شمارشی معادل با یک عدد ثابت است، این اعداد به طور پیش فرض از عدد صفر شروع می‌شوند و به ترتیب، یک واحد اضافه می‌شوند. مثلاً در تعریف قبل نام Sunday معادل با عدد صفر و نام Monday برابر یک و Saturday برابر ۶ است. اگر مایل باشید می‌توانید عدد دیگری را برای نام‌ها اختصاص دهید. در تعریف زیر نوع داده شمارشی برای ماه‌های یک سال میلادی را مشاهده می‌کنید. در این تعریف January معادل با عدد یک و February برابر دو است.

```
public enum MonthOfYear
{
    January = 1,
    February,
    March,
    April,
    May,
    June,
    July,
    August,
    September,
    October,
    November,
    December
}
```

سؤال: در قطعه برنامه بالا نام December معادل با چه عددی است؟

همان‌طور که با نوع داده‌های ساده و ابتدایی مانند int یا float کار می‌کنیم و می‌توانیم اعداد ثابت و یا متغیری از آن نوع را در برنامه مورد استفاده قرار دهیم دقیقاً همان عملیات را می‌توانیم با نوع داده شمارشی البته با کمی‌محدودیت انجام دهیم. در این قسمت با ذکر مثال با آنها آشنا می‌شویم.

۱ــ۱ــ۳ــ دسترسی به اعضای یک نوع داده شمارشی :

با نوشتن نام نوع داده شمارشی و سپس نام عضو که این دو با علامت نقطه از هم جدا می‌شوند، می‌توانیم به اعضا یا مقادیر یک نوع داده شمارشی دسترسی پیدا کنیم.

```
نام عضو . نوع داده شمارشی
   ↓          ↓
DayOfWeek . Saturday
```

شکل ۲ ــ۳ــ دسترسی به یک عضو نوع داده شمارشی

۲ــ۱ــ۳ــ تعریف یک متغیر از نوع داده شمارشی :

مانند متغیرهای ساده، متغیر نوع شمارشی قادر است فقط یکی از مقادیر نوع شمارشی را در خود جای دهد و به صورت شکل ۳ــ۳ تعریف می‌شود. اصول نام گذاری متغیرها را رعایت کرده و طبق روش کوهان شتری نام متغیر را می‌نویسیم.

```
; نام متغیر    نوع داده شمارشی
                  ↓          ↓
              DayOfWeek    holiday;
```

شکل ۳ــ۳ــ تعریف یک متغیر نوع داده شمارشی

۳ــ۱ــ۳ــ مقداردهی متغیرهای شمارشی :

مقداردهی متغیرهای نوع شمارشی معمولاً از طریق دستور انتساب به صورت شکل ۴ــ۳ انجام می‌شود.

```
; مقدار  =  نام متغیر
holiday = DayOfWeek . Sunday;
```

شکل ۴ــ۳ــ مقداردهی به یک متغیر نوع شمارشی

نکته

توجه داشته باشید مقداری که در متغیر نوع شمارشی قرار می‌گیرد باید با نوع آن مطابقت داشته باشد. مثلاً ذخیره عدد ۳ در متغیر holiday به صورت مستقیم امکان پذیر نیست و حتماً باید از تبدیل نوع استفاده کنید تا عضوی مطابق عدد ۳ در آن ذخیره گردد.

۴ــ۱ــ۳ــ نمایش مقدار یک عضو یا محتوای یک متغیر نوع شمارشی :

مانند متغیرهای ساده، از متد WriteLine() برای نمایش محتوای متغیری از نوع داده شمارشی و یا یک عضو از آن می‌توان استفاده کرد. شکل ۵ــ۳ را مشاهده کنید.

```
Console. writeLine ("My holiday is " + holiday);
Console. writeLine ("We should go to school on {0}", Day Of Week. Saturday);
```

نام عضو . نوع داده شمارشی

شکل ۵ــ۳ــ نمایش مقدار یک نوع داده شمارشی

کار در کارگاه ۱

مثال ۱ــ۳ : برنامه‌ای بنویسید که گروه سنی کاربر را با توجه به سن او طبق جدول ۱ــ۳ مشخص کند.

جدول ۱ــ۳ــ گروه سنی

محدوده سنی	نام
بالای ۱۸ سال	بزرگسالان (adult)
بالای ۱۵ سال	نیمه دوم نوجوانی(late teens)
بالای ۱۲ سال	ابتدای دوران نوجوانی (early teens)
بالای ۷ سال	کودکی(Children)
بالای ۳ سال	خردسالی(Infancy)

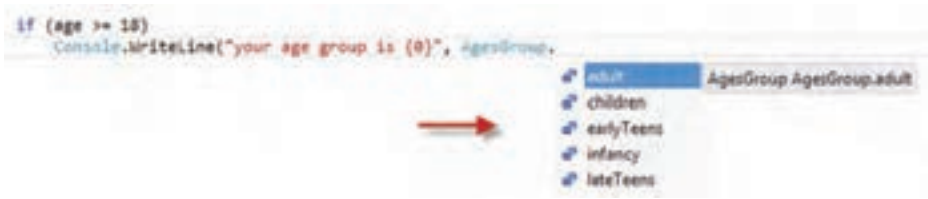
الگوریتم یا روش انجام کار : در این برنامه باید ابتدا سن کاربر سؤال شود و سپس با استفاده از دستورات if تودرتو، شرط‌های جدول یک به یک بررسی شود. اگر یک شرط برقرار شد گروه سنی نمایش داده شود. اگر کمی فکر کنیم می‌بینیم خواسته این برنامه، موضوع جدیدی نیست زیرا شبیه‌این کار را برای نمره یک دانش‌آموز قبلاً انجام داده‌اید. اما در این برنامه از نوع داده شمارشی استفاده می‌کنیم تا با مفهوم این نوع داده و روش استفاده از آن در یک برنامه آشنا شویم.

```
using System;
public enum AgesGroup
{
    adult, earlyTeens, late Teens, children, infancy
}
        class userAges Group
        {
            static void Main (string[] args)
            {
                int age = int. Parse (console. Readline());
                if (age > = 18)
                    Console. WriteLine ("your age group is {0}", AgesGroup. adult);
                else if (age > = 15)
                    Console. WriteLine ("your age group is {0}", AgesGroup. lateTeens);
                else if (age > = 12)
                    Console. WriteLine ("your age group is {0}", AgesGroup. earlyTeens);
                else if (age > = 7)
                    Console. WriteLine ("your age group is {0}", AgesGroup. children);
                else if (age > = 3)
                    Console. WriteLine ("your age group is {0}", AgesGroup. infancy);
            }
}
```

برنامه ۱ــ۳ــ استفاده از نوع داده شمارشی برای تشخیص گروه سنی

۱ــ در هنگام نوشتن برنامه، از منوی IntelliSense برای نوشتن سریع نوع داده شمارشی مانند شکل ۶ــ۳ کمک بگیرید. در برنامه ۱ــ۳، نوع داده AgesGroup با پنج عضو تعریف شده است، هر

یک از اعضا، معادل یک عدد ثابت است. adult معادل صفر و earlyTeens معادل عدد یک و الی آخر. . .



شکل ۶ــ۳ــ مقداردهی از نوع داده شمارشی

۲ــ در هنگام اجرای برنامه ۱ــ۳، اعداد مختلفی را به عنوان سن کاربر وارد کنید تا خروجی برنامه را مشاهده کنید.

۳ــ دستوراتی به ابتدای برنامه اضافه کنید تا نام کاربر سؤال شود و در پایان برنامه همراه با گروه سنی وی چاپ گردد.

سؤال: در مثال ۱ــ۳ به‌جای سن کاربر سال تولد او را دریافت کرده و گروه سنی او را مشخص کنید.

مثال ۲ــ۳ : برنامه‌ای بنویسید که مقام یک بازیکن را برحسب تعداد سکه‌ای که در اختیار دارد طبق جدول ۲ــ۳ مشخص کند.

جدول ۲ــ۳ــ درجه یک بازیکن

تعداد سکه	مقام بازیکن
تا ۵۰۰۰	معمولی
تا ۱۰۰۰۰	ارشد
تا ۱۵۰۰۰	معاون
از ۱۵۰۰۰ به بالا	رئیس

الگوریتم و روش انجام کار : در این برنامه باید ابتدا تعداد سکه بازیکن سؤال شود و سپس با استفاده از دستورات if تودرتو، شرط‌های جدول یک به یک بررسی شود. اگر یک شرط برقرار شد درجه بازیکن نمایش داده شود.

در برنامه ۲ــ۳، نوع داده PlayerRank با چهار عضو تعریف شده است، هر یک از اعضا، معادل یک عدد ثابت است. Leader معادل صفر و Co_Leader معادل عدد یک و الی آخر. . . .

```
using System;
public enum PlayerRank
{
    Leader, Co-Leader, Elder, Member
}
class playerRankDemo
{
    static void Main()
    {
        PlayerRank player;          ← تعریف متغیر از نوع داده شمارشی
        Console. Write ("Enter Player coins: ");
        string input = Console. ReadLine();
        long coins = long. Parse (input);

        if (coins <10000)
            player = PlayerRank. Member;
        else
            if (coins <15000)
                player = PlayerRank. Elder;
            else
                if (coins <20000)
                    player = PlayerRank. Co-Leader;
                else
                    player = PlayerRank. Leader;
        Console. WriteLine ("The player Rank: {0}", player);
    }
}
```

برنامه ۲ــ۳ــ استفاده از نوع داده شمارشی برای تشخیص مقام یک بازیکن

۱ــ در هنگام نوشتن برنامه، از منوی IntelliSense برای نوشتن سریع نوع داده شمارشی مانند شکل ۷ــ۳ کمک بگیرید.

شکل ۷ــ۳ــ استفاده از نوع داده شمارشی

۲ــ در هنگام اجرای برنامه ۲ــ۳، اعداد مختلفی را به عنوان تعداد سکه وارد کنید تا خروجی برنامه را مشاهده کنید.

۳ــ دستوراتی به ابتدای برنامه اضافه کنید تا نام بازیکن سؤال شود و در پایان برنامه همراه با مقام وی چاپ گردد.

۴ــ دستوراتی به انتهای برنامه ۲ــ ۳ اضافه کرده تا پس از نمایش مقام بازیکن، تعداد الماس[۱] وی را سؤال کند. (قبل از دریافت تعداد الماس، پیام مناسب چاپ کند.)

۵ ــ دستوراتی به انتهای برنامه اضافه کنید که اگر تعداد الماس وی بیش از ۵۰۰۰ بود به بازیکن پیشنهاد داده شود که آیا مایل هستید با خرج کردن (از دست دادن) تعداد ۵۰۰۰ الماس وضعیت و مقام خود را یک پله افزایش دهید؟ اگر کاربر مایل بود و پاسخ Yes داد در این صورت مقام وی یک پله بالا رود و در غیر این صورت در همان وضعیت باقی بماند. در انتهای برنامه مقام جدید، تعداد سکه و الماس باقیمانده وی با رنگ‌های مناسب نشان داده شود.

۲ــ۳ــ استفاده از نوع داده‌های شمارشی آماده در کتابخانه .NET

انواع مختلفی از نوع داده‌های شمارشی آماده در کتابخانه .NET وجود دارد، از جمله نوع شمارشی ConsoleColor که بارها برای تعیین رنگ زمینه و قلم صفحه کنسول از آن استفاده کردید. در واقع به جای حفظ کردن تعدادی عدد، از نام رنگ‌ها که قبلاً در ذهن ما نقش بسته است، استفاده می‌کنیم. مثلاً برای اینکه پیامی به رنگ زرد بر روی زمینه آبی نوشته شود، از دستورات زیر

۱ ــ Gems

استفاده می‌کنید :

```
Console. Background Color = ConsoleColor. DarkBlue;
Console. Foreground Color = ConsoleColor. Yellow;
Console. WriteLine ("Using colors in console mode.");
```

۳ــ ۳ــ شئ[۱]

در زبان محاوره‌ای، با شنیدن کلمه شئ، به یاد یک جسم بی‌جان می‌افتیم. به عنوان مثال، در محیط مدرسه، روزانه با اشیایی نظیر کتاب، دفتر و توپ فوتبال، سروکار داریم. تصور از کلمه شئ می‌تواند کمی توسعه یابد به‌طوری که موضوعات دیگر نظیر کارنامه دانش‌آموز و یا فاکتور خرید یک کالا را نیز به عنوان یک شئ در نظر بگیریم.

هر شئ دارای ویژگی‌هایی است که آن را از اشیای دیگر متمایز می‌سازد. مثلاً یک توپ فوتبال دارای ویژگی‌هایی مانند رنگ، طرح و اندازه است که آن را از توپ فوتبال دیگر متمایز می‌سازد و یا در وضعیت ساکن و یا در حال حرکت است.

بر روی هر شئ نیز عملیاتی می‌توان انجام داد. مثلاً در مورد توپ فوتبال می‌توان عمل شوت زدن، پرت کردن و یا گرفتن توپ را انجام داد.

دانش‌آموز را نیز می‌توانیم یک موجود در نظر بگیریم که دارای ویژگی‌هایی مانند نام و نام‌خانوادگی، نام پدر، شماره ملی است که وی را از بقیه دانش‌آموزان متمایز می‌سازد. همچنین دارای رفتارهای مختلف است. وقتی از وی نامش را سؤال می‌کنند پاسخ می‌دهد، وقتی دوستان و آشنایان را می‌بینند سلام می‌کند. در امتحان شرکت می‌کند، به سؤالات پاسخ می‌دهد و نمره می‌گیرد، خوشحال می‌شود، می‌خندد.

در زندگی روزمره اشیاء و موجودات مختلف با یکدیگر در ارتباط و تعامل هستند. توپ فوتبال توسط دانش‌آموزی شوت می‌شود، توپ به حرکت درآمده و سرعت می‌گیرد و وضعیت آن تغییر می‌کند. دانش‌آموز دیگری برای گرفتن توپ تلاش می‌کند و در اثر باخت بازی

---

۱ ــ Object

ناراحت شده و حتی گریه می‌کند.

دانشمندان رشته ریاضی و کامپیوتر، سعی کرده‌اند که زبان‌های برنامه نویسی طراحی کنند که در آنها، برنامه‌نویس بتواند ویژگی‌ها و رفتارهای اشیاء و موجودات زندگی واقعی را مدل سازی کند تا بتواند برنامه‌ای برای حل مسائل در ارتباط با آنها بنویسد.

زبان‌های برنامه‌نویسی که در آنها امکان تعریف ویژگی‌ها و رفتارهای اشیاء و موجودات فراهم شده است، همچنین اجازه می‌دهند اشیایی برمبنای ویژگی‌های تعریف شده، ایجاد شوند و با یکدیگر ارتباط و نسبت به هم واکنش داشته باشند، زبان‌های شئ گرا[1] نامیده می‌شود.

زبان‌های C++، جاوا و C# از جمله زبان‌های شئ گرا هستند، برنامه‌ای که به این زبان‌ها نوشته می‌شود و اجرا می‌گردد، در واقع از تعدادی شئ تشکیل شده است که با یکدیگر در ارتباط و تعامل هستند.

کلمه شئ در این زبان‌ها، نه تنها اشیای بی‌جان، بلکه موجودات جان‌دار نظیر دانش‌آموز، کارمند را نیز در بر می‌گیرد (شاید بهتر باشد به جای کلمه شئ، کلمه موضوع را به کار ببریم که می‌تواند به هر موجود یا اشیایی اشاره کند).

با توجه به مطالب گفته شده، یک شئ شامل تعدادی ویژگی، وضعیت، رفتار و عملیات است که آن را از اشیای دیگر متمایز می‌سازد.

مثال ۳ــ۳ : کارنامه یک دانش‌آموز را به عنوان یک شئ در نظر می‌گیریم، ویژگی‌های متمایز کننده و عملیات و رفتارهای انجام شده بر روی آن می‌تواند چنین باشد :

مقادیری نظیر نام و نام خانوادگی، نام دروس و نمرات هر یک از آنها، ازجمله ویژگی‌هایی است که یک کارنامه را از کارنامه دیگری جدا و قابل تشخیص می‌کند و برای یک دانش‌آموز و مدرسه اهمیت دارد. وضعیت یک کارنامه می‌تواند قبولی، ردی یا مشروطی باشد. عملیات متداولی که بر روی یک کارنامه صورت می‌گیرد، وارد کردن نمرات در کارنامه، رتبه بندی، چاپ کارنامه و استخراج معدل است که توسط دفتردار مدرسه انجام می‌شود. برای حل یک مسئله به روش شئ گرا، تشخیص اشیاء و ارتباط آنها با یکدیگر بسیار اهمیت دارد.

ویژگی‌ها و وضعیت یک شئ به وسیله تعدادی متغیر که فیلد نامیده می‌شوند، مشخص می‌شود و رفتارهای اشیاء در قالب متدها تعریف می‌گردند. بنابراین محل و مکان تعریف فیلدها و متدهای یک شئ در داخل یک کلاس است (شکل ۸ــ۳).

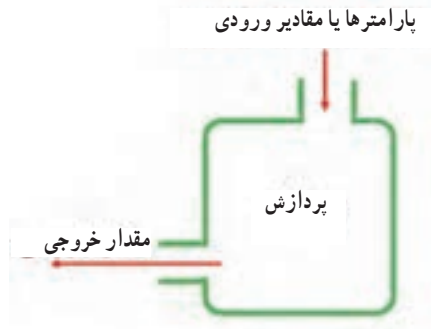
۱ــ Object Oriented Language

شکل ۸ــ۳

با توجه به شکل، می‌توان گفت که **کلاس** تعریف مشخصات، وضعیت و رفتارهای یک شئ را در بردارد و نوع شئ را مشخص می‌کند. بنابراین بهتر است نام کلاس، مطابق با نام شئ باشد که در حال تعریف آن هستیم.

۱ــ۳ــ۳ــ متد :

همان‌طور که می‌دانید متد، مجموعه‌ای از دستورات است که برای انجام یک عمل خاص و حل یک مسئله کوچک به کار می رود. به‌طورکلی متد را می‌توان مانند یک دستگاه در نظر گرفت، که از یک طرف موادی وارد آن می‌شود و از طرف دیگر موادی تغییر یافته، از آن خارج می‌شود. این مواد معمولاً داده‌ها هستند که بر روی آنها یک مرحله پردازش صورت می‌گیرد.



شکل ۹ــ۳ــ پردازش پارامترها

نکته

محل تعریف هر متد در داخل یک کلاس است.

متد جزیی از یک کلاس است. برنامه‌هایی که تاکنون نوشته‌ایم دارای یک کلاس و متد Main بوده‌اند. شکل و ساختار تعریف متد ()Main را در شکل ۱۰ــ۳ مشاهده می‌کنید :

```
static void Main (String[] args)  ← عنوان
{
                                  ← بدنه متد
}
```

شکل ۱۰ــ۳ــ عنوان و بدنه متد Main

تعریف متد ()Main و یا هر متد دیگر، از دو قسمت تشکیل می‌شود :

۱ــ عنوان[۱] متد

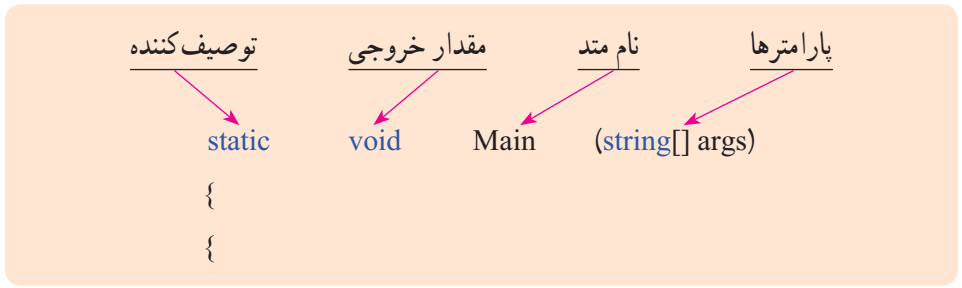
۲ــ بدنه متد

همان‌طور که با مشاهده شناسنامه یک فرد، اطلاعات مهمی‌در مورد مشخصات فردی وی بدست می آوریم، از روی خط عنوان یک متد نیز، اطلاعات اساسی درمورد متد و طریقه استفاده از آن را متوجه می‌شویم. به خط اول تعریف متد، عنوان متد می گویند. در قسمت عنوان متد، مشخصاتی مانند، نام متد، نوع داده خروجی، لیست داده‌های ورودی (پارامترها[۲]) و همچنین روش دسترسی[۳] به متد و نحوه استفاده از آن معین و تعریف می شود. بنابراین خط عنوان یک متد، بسیار بسیار مهم است و حاوی اطلاعات اساسی در مورد روش استفاده از متد است. در شکل ۱۰ــ۳ خط عنوان متد ()Main که در ویژوال استودیو مشاهده می‌کنید، نشان داده شده است. حال به شرح هر یک از قسمت‌های آن می‌پردازیم.

۱ــ Heading

۲ــ Parameters

۳ــ Access



شکل ۱۱ــ۳ــ معرفی خط عنوان متد Main()

بدنه متد همان دستوراتی است که در داخل متد نوشته می‌شود و در برنامه‌های قبلی بارها در متد Main نوشته‌اید و با این دستورات آشنا هستید.

۱ــ۱ــ۳ــ۳ــ توصیف کننده[۱] : خط عنوان یک متد با کلمه‌ای شروع می‌شود که روش ایجاد و محدوده دسترسی به متد را مشخص می‌کند. به عنوان مثال توصیف کننده static، روش ایجاد متد را مشخص می‌کند و نشان می‌دهد که به محض اجرای برنامه، چنین متدی ساخته شده و قابل استفاده است.

متد Main() نیز همواره باید از نوع static باشد اگر توصیف‌کننده static را ننویسید، مترجم در ترجمه برنامه خطا می‌دهد. تمام متدهای آماده‌ای که تاکنون استفاده کرده‌اید نظیر ReadLine() نیز از نوع استاتیک هستند که در کلاس Console از قبل تعریف شده‌اند. استفاده از متدهای static ساده است.

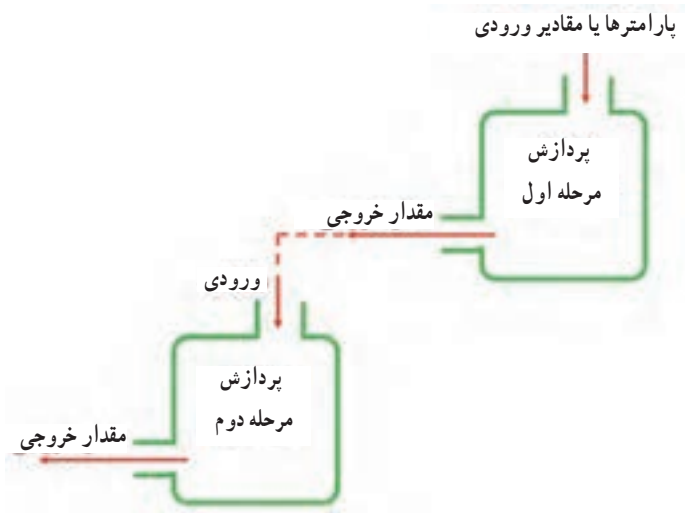
توصیف کننده‌های دیگری نظیر private و protected وجود دارند که به وسیلهٔ آنها می‌توان محدوده دسترسی به متد را به یک کلاس محدود کرد و یا با استفاده از توصیف کننده public محدوده دسترسی به یک متد را بسیار گسترده و در واقع، بدون محدودیت معین کرد. متدهای ReadLine() و WriteLine() با توصیف‌کننده public تعریف شده‌اند و به همین دلیل است که می‌توانید در هر برنامه‌ای (هر کلاسی) از آنها استفاده کنید.

نکته

اگر هیچ توصیف کننده‌ای برای تعیین محدوده دسترسی یک متد ذکر نشود، نوع دسترسی private به‌طور پیش فرض برای متد در نظر گرفته می‌شود.

۱ــ Modifier

۲ــ۱ــ۳ــ۳ــ مقدار خروجی یا نوع داده برگشتی[۱] : در خط عنوان متد، پس از توصیف‌کننده‌ها، مقدار خروجی یا نوع داده برگشتی متد را باید مشخص کنید. **منظور از مقدار خروجی، آنچه که روی صفحه نمایش چاپ می شود نیست** بلکه مقداری است که از طریق اجرای متد قابل دریافت است و می‌توان آن را در یک متغیر ذخیره کرد و یا به متد دیگری برای ادامه پردازش ارسال کرد. شکل ۱۲ــ۳ را ملاحظه کنید.



شکل ۱۲ــ۳ــ استفاده از خروجی یک متد به عنوان ورودی متد دیگر

به عنوان مثال، می‌خواهیم شعاع دایره را از ورودی گرفته و مساحت آن را محاسبه کنیم. بدین‌منظور باید ابتدا از متد ReadLine استفاده کرده و عدد را دریافت کنیم، عدد دریافت شده به صورت string است، پس باید با استفاده از متد Parse() عدد را به نوع داده اعشاری تبدیل کنیم.

```
string numStr=Console.ReadLine();
float num = float.Parse(numStr);
```

ممکن است متدی مقدار برگشتی یا خروجی نداشته باشد مانند متد WriteLine()، در این صورت نوع برگشتی آن void یا بدون مقدار خروجی است. معمولاً متد Main() نیز، بدون مقدار

۱ــ Return Type

برگشتی (void) است[1].

در مقابل متد WriteLine()، متد ReadLine() را درنظر بگیرید که رشته‌ای را از کاربر دریافت می‌کند و آن را به برنامه تحویل می‌دهد. مقدار برگشتی این متد، یک رشته است و معمولاً آن را در یک متغیر از نوع رشته‌ای ذخیره می‌کنیم.

نکته

ممکن است یک متد، مقدار خروجی و یا مقدار ورودی نداشته باشد.

۳ــ۱ــ۳ــ۳ــ نام متد : در خط عنوان متد، پس از تعیین نوع برگشتی، باید نام متد را تعیین کنیم. در نام‌گذاری متد باید مانند نام متغیرها، اصول نام گذاری شناسه‌ها را رعایت کنیم. معمولاً نام متد به صورت یک فعل امری نام‌گذاری می‌شود. این نام بهتر است بیان کننده نوع کاری باشد که متد، قصد انجام آن را دارد. برای مثال متد Clear برای پاک کردن صفحه نمایش و متد Write برای نوشتن روی صفحه نمایش است.

۴ــ۱ــ۳ــ۳ــ لیست پارامترهای ارسالی : در خط عنوان متد، پس از تعیین نام، باید پارامترها یا داده‌هایی که از طرف برنامه به متد داده می‌شود (اصطلاحاً به متد ارسال می‌شود) را مشخص کنیم. مثلاً برای متد Write پارامتر ورودی می‌تواند یک رشته باشد که باید روی صفحه کنسول نمایش داده شود. بعضی از متدها نیز پارامتر ورودی ندارند مانند متد ReadLine().

به عنوان مثال، خط عنوان متد WriteLine() را بررسی می‌کنیم :

```
public static void WriteLine (string value)
```

خروجی — نوع داده ورودی

با توجه به خط عنوان، نام متد WriteLine است و ورودی آن یک رشته مانند value است. این متد خروجی ندارد و با دسترسی وسیع و از نوع استاتیک است.

سؤال: با توجه به خط عنوان متد WriteLine()، توصیف کننده‌های آن کدام است؟

۱ــ البته خروجی متد Main() می‌تواند عدد صحیح int نیز باشد.

۵ــ۱ــ۳ــ۳ــ روش استفاده یا فراخوانی[۱] متد : باید بتوانیم از متد در برنامه استفاده کنیم و یا به اصطلاح آن را فراخوانی کنیم. چنانچه متدی از نوع استاتیک باشد فراخوانی آن بسیار ساده است به شرط اینکه خط عنوان آن را بدانیم. برای اطلاع از خط عنوان متدهای آماده Net. می‌توانید به راهنمای آنها مراجعه کنید. پس از اطلاع از عنوان متد، کافی است فضای نامی و نام کلاس را قبل از نام متد ذکر کنید. این سه قسمت با علامت نقطه از یکدیگر جدا می‌شوند. اگر متد دارای پارامتر است باید مقدار پارامترها را نیز در داخل پرانتز معین کنید و در انتها علامت نقطه ویرگول را بنویسید.

فضای نامی . نام کلاس . نام متد ; مقدار پارامترها

فرض کنید می‌خواهیم صدایی با فرکانس ۲۰۰۰ هرتز به مدت ۱ ثانیه ایجاد کنیم. در کتاب برنامه‌سازی۱ از متد ()Beep برای ایجاد صدا استفاده کردید. این متد صدا را با فرکانس معین در مدت زمان مشخص تولید می‌کند. خط عنوان این متد به صورت زیر است :

```
public static void Beep(int frequency, int duration)
```

این متد به صورت استاتیک و در کلاس Console تعریف شده است و خروجی ندارد. این کلاس در فضای نامی System قرار دارد، بنابراین فراخوانی متد ()Beep چنین است :

```
System.Console.Beep(2000,1000)
```

سؤال: در چه صورت می‌توانیم دستور بالا، را به صورت زیر خلاصه کنیم؟

```
Console.Beep(2000,1000)
```

فرض کنید می‌خواهیم نام کاربر را از ورودی دریافت کنیم. بارها از متد RedLine برای دریافت از ورودی استفاده کرده‌اید. خط عنوان این متد به صورت زیر است :

```
public static string ReadLine( )
```

این متد ورودی ندارد و خروجی آن از نوع string است و به صورت استاتیک و در کلاس

۱ــ Method call

Console تعریف شده است و در فضای نامی System قرار دارد، بنابراین فراخوانی متد ReadLine() چنین است :

```
System.Console.ReadLine( )
```

سؤال: در چه صورت می‌توانیم دستور بالا، را به صورت زیر خلاصه کنیم؟

```
Console.ReadLine()
```

با اجرای هر یک از دستورات بالا، نام کاربر از ورودی دریافت می‌شود ولی برای استفاده در جایی ذخیره نمی‌شود. برای استفاده از نام کاربر، باید خروجی متد را در یک متغیر ذخیره کنیم و یا به متد WriteLine() برای چاپ بر روی صفحه نمایش ارسال کنیم. برای ذخیره نام کاربر در یک متغیر می‌توانیم چنین دستوری بنویسیم :

```
string userName = Console.ReadLine( );
```

و همچنین برای نشان دادن نام کاربر بر روی صفحه نمایش، چنین دستوری می‌نویسیم :

```
Console.WriteLine(Console.ReadLine( ));
```

سؤال: چرا متغیر userName از نوع string تعریف شده است؟

خودآزمایی فصل سوم

الف) در سؤالاتِ چند گزینه‌ای زیر پاسخ صحیح را انتخاب نمایید.

۱ــ چه چیز یک شئ را از اشیا دیگر متمایز نمی‌کند؟

الف) مشخصه ب)رفتار ج)عملیات د)توصیف کننده

۲ــ کدام گزینه در مورد متد صحیح نیست؟

الف) محل تعریف متدهای یک شئ داخل کلاس است.

ب) نحوه دسترسی به متد در خط عنوان متد تعیین می‌شود.

ج) یک متد می‌تواند پارامتر ورودی نداشته باشد.

د) اگر محدوده دسترسی به متد تعیین نشود به طور پیش فرض public در نظر گرفته می‌شود.

۳ــ کدام گزینه در مورد نوع داده شمارشی صحیح است؟

الف) محل قرارگیری تعریف نوع داده شمارشی در متد Main است.

ب) شماره معادل با اولین عضو نوع داده شمارشی، عدد یک است.

ج) برای جدا کردن نام‌ها در نوع داده شمارشی، علامت , به کار می‌رود.

د) نوع دسترسی به یک نوع داده شمارشی معمولاً private به کار می‌رود

ب) جاهای خالی را با عبارات مناسب پرکنید.

۴ــ مجموعه‌ای از چند نام دلخواه که حالت‌ها و مقادیر مختلف یک موضوع را نشان می‌دهند ..... نام دارد.

۵ــ اگر یک متد مقداری برنمی‌گرداند، نوع خروجی آن را ..... تعیین می‌کنیم.

۶ــ کلمه‌ای که خط عنوان متد با آن شروع شده و محدوده دسترسی به متد را تعیین می‌کند ..... نام دارد.

ج) به سؤالات زیر پاسخ دهید.

۷ــ ویژگی‌ها و رفتار و عملیات قابل انجام بر روی هر یک از اشیای زیر را تعیین کنید.

کتاب، تلویزیون، دستگاه بازی

۸ــ با یک مثال نشان دهید که خروجی یک متد می‌تواند ورودی یک متد دیگر باشد.

۹ــ روش ایجاد اکثر متدهای آماده کتابخانهNET. چیست؟

۱۰ــ چرا روش ایجاد متد ()Main باید static باشد؟

۱۱ــ اشکال برنامه زیر چیست؟ با یک تغییر ساده در برنامه اشکال را برطرف کنید.

```
using System;
namespace checkError
{
  class Program
  {
    static void Main(string[] args)
    {
       enum Sesean
    { spring, summer, fall, winter  }

      Console.ReadKey();
    }
  }
}
```

فعالیت

متن زیر از MSDN با موضوع داده‌های شمارشی برداشت شده است.

enum

The enum keyword is used to declare an enumeration, a distinct type that consists of a set of named constants called the enumerator list.

Usually it is best to define an enum directly within a namespace so that all classes in the namespace can access it with equal convenience. However, an enum can also be nested within a class or struct.

By default, the first enumerator has the value 0, and the value of each successive enumerator is increased by 1. For example, in the following enumeration, Sat is 0, Sun is 1, Mon is 2, and so forth.

```
enum Days {Sat, Sun, Mon, Tue, Wed, Thu, Fri};
```

Enumerators can use initializers to override the default values, as shown in the following example.

```
enum Days {Sat=1, Sun, Mon, Tue, Wed, Thu, Fri};
```

In this enumeration, the sequence of elements is forced to start from 1 instead of 0. However, including a constant that has the value of 0 is recommended. For more information, see Enumeration Types (C# Programming Guide).

Every enumeration type has an underlying type, which can be any integral type except char. The default underlying type of enumeration elements is int. To declare an enum of another integral type, such as byte, use a colon after the identifier followed by the type, as shown in the following example.

```
enum Days: byte {Sat=1, Sun, Mon, Tue, Wed, Thu, Fri};
```

The approved types for an enum are byte, sbyte, short, ushort, int, uint, long, or ulong.

واژگان و اصطلاحات انگلیسی فصل سوم

ردیف	واژه انگلیسی	معنی به فارسی
۱	Access	
۲	Behavior	
۳	Enumerated Type	
۴	Gem	
۵	Heading	
۶	Member	
۷	Method Call	
۸	Modifier	
۹	Object Oriented Language	
۱۰	Parameter	
۱۱	Property	
۱۲	Return Type	