فصل اول: یادآوری و تکمیل مطالب

راه های ایجاد برنامه به زبان C#

۱. استفاده از ویرایشگرهای متنی مانند notepad:

ویرایشگرهای متنی قابلیت اجرای برنامه را ندارند بنابراین بعد از نوشتن کد برنامه باید آنرا با استفاده از مترجم زبان #c یعنی csc.exe ترجمه کنیم. کدنویسی در این محیط وقت گیر است و قواعد زبان #C را باید بدانیم.

۲. استفاده از محیط توسعه متمرکز(IDE)

الف- محیط Visual Studio Express 2012(VS)

- محیطی بسیار قدرتمند برای نوشتن، اجرا و عیب یابی برنامه
- استفاده از لیست هوشمند (intellisense) که باعث افزایش سرعت تایپ برنامه و املای صحیح دستورات می شود.
- رنگ بندی کدهای برنامه که باعث افزایش خوانایی برنامه می شود.
- استفاده از insert snippet برای نمایش ساختار یک دستور که با کلیک راست روی صفحه ظاهر می‌شود.

ب - محیط #develop (SharpDevelop)

- یک IDE رایگان، کم حجم و متن باز (open source) با امکانات پیشرفته

یادآوری دستورات ورودی و خروجی

دستورات ورودی: برای دریافت داده ها از کاربر استفاده می شوند.

- دستور ReadLine()

نکته: خروجی این متد از نوع رشته ای است بنابراین هنگام دریافت اعداد توسط این دستور باید رشته عددی به عدد واقعی تبدیل شود تا بتوان روی آن محاسبات لازم انجام شود. متد parse() برای این منظور استفاده می شود.

مثال:

```
int x=int.Parse(Console.ReadLine())
```

دستورات خروجی: برای نمایش پیغام، محتوای یک متغیر یا نتیجه یک عبارت بکار می روند.

- WriteLine(): بعد از نمایش خروجی مکان نما به ابتدای سطر بعد منتقل می شود
- Write() : بعد از نمایش خروجی مکان نما در همان سطر باقی می ماند.

الگوی جایگذاری در رشته

در هنگام نمایش یک رشته می توان اعداد یا محتوای متغیرها را با استفاده از این الگو در داخل رشته جایگذاری کرد به صورت زیر:

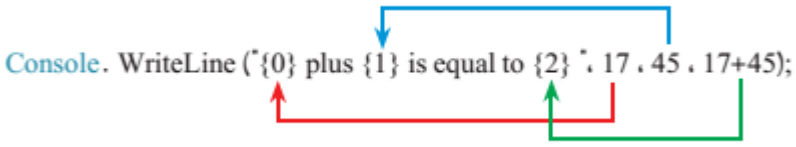
{الگوی نمایش:عدد تراز,شماره}

«شماره»: ترتیب نمایش متغییرها را می توان با آن مشخص کرد و از صفر شروع می شود.

«عدد تراز»: مقدار فضایی که برای نمایش محتوای متغیر در نظر گرفته می شود. در صورتی که «عدد تراز» مثبت باشد مقدار خروجی در سمت راست فضای در نظر گرفته شده (راست چین) و در صورتی که منفی باشد در سمت چپ فضا نمایش داده می شود(چپ چین).

«الگوی نمایش»: برای نمایش خروجی با قالب های مختلف بکار می رود. (جدول پیوست کتاب)

مثال: حاصلجمع ۱۷ و ۴۵ را با استفاده از الگوی جایگذاری نمایش دهید.



مثال: میانگین اعداد ۱۲,۷۵و ۱۹و ۱۳,۲۵ را با استفاده از الگوی جایگذاری و با دو رقم اعشار نمایش دهید.

```
Float aver=(12.75+19+13.25)/3;
Console.writeline("average is {0,10:F}",aver);
```

دستور بالا باعث نمایش معدل در فضایی به اندازه ۱۰ کاراکتر و در سمت راست فضا (راست چین) و بصورت دو رقم اعشار می‌شود.

یادآوری دستورات شرطی (if, switch, ?)

برای تصمیم گیری در اجرای دستورات استفاده می شوند.

الف- دستور if: در جلوی دستور if یک عبارت منطقی ساده و یا مرکب به عنوان شرط نوشته می‌شود، درصورت درست بودن شرط، دستور یا دستورات بعد از این عبارت اجرا می‌شود، در غیر این صورت کنترل برنامه به دستور بعد از if منتقل می‌شود.

```
If (شرط)
{
دستور یا دستوراتی که در صورت درست بودن شرط اجرا می شوند
}
```

ب- دستور if-else : در جلوی دستور if یک عبارت منطقی ساده و یا مرکب به عنوان شرط نوشته می‌شود، درصورت درست بودن شرط، دستور یا دستورات بعد از این عبارت اجرا می‌شود، در غیر این صورت دستور یا دستورات پس از else اجرا می‌شود.

```
If (شرط)
{
دستور یا دستوراتی که در صورت درست بودن شرط اجرا می شوند
}
else
{
دستور یا دستوراتی که در صورت نادرست بودن شرط اجرا می شوند
}
```

مثال : برنامه زیر معتبر بودن نمره‌ی دریافتی را بررسی می‌کند.

```
Console. Write ("Enter a mark:");
float number= float. parse (Console. ReadLine ());
if ((number<0) | | (number>20))
    Console. WriteLine ("Invalid Mark!");    //نامعتبر
else
    Console. WriteLine ("Mark is in valid range");    //معتبر
```

عملگر سه تایی ؟

بجای استفاده از دستور if – else می توان از این عملگر به شکل زیر استفاده کرد:

مقدار دوم : مقدار اول ? (عبارت منطقی)

کامپیوتر در هنگام برخورد با عملگر سه تایی ابتدا حاصل عبارت منطقی را محاسبه میکند، اگر حاصل عبارت منطقی «درست» باشد، نتیجه این عملگر برابر «مقدار اول» خواهد بود و در صورتی که حاصل عبارت منطقی، نادرست باشد، حاصل این عملگر برابر «مقدار دوم» است.

مثال: برنامه ای بنویسید که با دریافت معدل دانش آموز، وضعیت قبولی یا مردودی وی را نمایش دهد.

```
float aver;
string result;
aver=float.Parse(Console.ReadLine());
result=(aver>=10) ? "ghabol":"mardod";
Console.WriteLine(result);
```

ج- دستور if-else if : از این دستور زمانی استفاده می شود که بخواهیم چندین مقایسه در برنامه انجام دهیم و بخاطر پیچیدگی هایی که در برنامه ایجاد می کند توصیه شده است بجای آن از دستور switch استفاده شود.

د- دستور switch : برای مواقعی که بخواهیم حالت های مختلف یک عبارت را بررسی کرده و بر اساس آن دستورات را اجرا کنیم از دستور switch استفاده می کنیم.

```
Switch (متغیر یا عبارت مورد مقایسه)
{
Case    مقدار ۱ :
        یک یا چند دستور;
        Break;
Case    مقدار ۲ :
        یک یا چند دستور;
        Break;
…
default:
        یک یا چند دستور;
}
```

مثال: برنامه ای بنویسید که یک عدد معادل فصل‌های سال (۱ تا ۴) از ورودی دریافت کرده و متناسب با آن رنگ زمینه صفحه را تغییر دهید. اگر عدد وارد شده در محدوده ی فصل های سال نباشد پیغام مناسب صادر شود.

```
Console.WriteLine("Enter Number(1-4):");
int f =int.Parse(Console.ReadLine());
switch (f)
{
    case 1:
        Console.BackgroundColor = ConsoleColor.Green;
        Console.Clear();
        Console.WriteLine("Spring");
        break;
    case 2:
        Console.BackgroundColor = ConsoleColor.Blue;
        Console.Clear();
        Console.WriteLine("Summer");
        break;
    case 3:
        Console.BackgroundColor = ConsoleColor.Yellow;
        Console.Clear();
```

```
            Console.WriteLine("Fall");
            break;
        case 4:
            Console.BackgroundColor = ConsoleColor.White;
            Console.Clear();
            Console.WriteLine("Winter");
            break;
        default:
            Console.BackgroundColor = ConsoleColor.Red;
            Console.Clear();
            Console.WriteLine("Invalid Number!");
            break;
    }
```

متد TryParse()

اگر کاربر در هنگام ورود داده های عددی، کاراکتری غیر از عدد وارد کند، متد parse() در تبدیل آن به عدد، دچار خطا شده و برنامه به طور ناگهانی قطع می شود و خطایی ظاهر می شود. برای جلوگیری از این مشکل از متد TryParse() استفاده می شود.

TryParse(رشته عددی, out متغیر)

در این متد اگر تبدیل انجام شود مقدار تبدیل شده را به ورودی دوم خود می ریزد و اگر موفق به تبدیل نشود با تولید مقدار منطقی false، برنامه نویس می تواند برنامه را کنترل کند.

مثال: دریافت معدل و تشخیص وضعیت دانش آموز(قبولی یا مردودی)

```
float n;
Console.Write("Enter average:");
string str = Console.ReadLine();
if (float.TryParse(str, out n))
{
    str = (n >= 10) ? "ghabol" : "mardod";
    Console.WriteLine(str);
}
else
    Console.WriteLine("Input Error.");
```

ثابت (constant)

- ثابت، مکانی است در حافظه برای نگهداری داده ها که مقدار آن در طول اجرای برنامه ثابت است.
- برای تعریف شناسه یا نام ثابت از کلمه کلیدی const استفاده می‌شود.
- معمولاً نامی‌که برای اعداد ثابت تعریف می‌شود با حروف بزرگ نوشته می‌شود تا در برنامه مشخص باشد که این شناسه، یک مقدار ثابت است.
- نحوه تعریف ثابت‌ها، مانند تعریف متغیرها می‌باشد با این تفاوت که در ابتدای تعریف آنها کلمه const قرار دارد.

نحوه ی تعریف ثابت:

const نوع داده نام ثابت=مقدار;

مثال:

```
const double PINUMBER=3.14;
```

دلایل استفاده از ثابت

- ویرایش ساده تر برنامه در آینده
- افزایش خوانایی برنامه

مثال: با دریافت شعاع دایره، محیط و مساحت آنرا محاسبه نمایید.

```
int r;
const double PINUMBER = 3.14;
Console.Write("Enter Radius:");
string str = Console.ReadLine();
if(int.TryParse(str,out r))
{
    double m1 = 2 * PINUMBER * r;
    double m2 = PINUMBER * r * r;
    Console.WriteLine("mohit:{0}, masahat:{1}",m1,m2);
}
else
{
    Console.WriteLine("input error");
}
```

یادآوری دستورات تکرار یا حلقه (for, while, do-while)

در مواقعی که می خواهیم یک یا چند دستور، بیش از یک بار انجام شوند، به جای اینکه آنها را چندین بار تایپ و یا کپی کنیم، می توانیم از دستورات تکرار یا حلقه استفاده نماییم.

اجزاء حلقه

- متغیر شمارنده: متغیر کمکی برای شمارش تعداد دفعات تکرار دستورات حلقه که قبل از شروع حلقه مقدار اولیه می‌گیرد.
- گام افزایش یا کاهش: مقداری که در هر بار اجرای حلقه به متغیر شمارنده افزوده یا کاسته می شود.
- شرط خاتمه: شرطی که با توجه به آن پایان حلقه مشخص می شود.
- بدنه حلقه: دستورات داخل حلقه

الف- دستور for: برای اجرای یک یا چند دستور به تعداد دفعات معین بکار می‌رود.

شکل استفاده:

```
for (int متغیر شمارنده = مقدار اولیه; شرط خاتمه; گام افزایش)
 {
       بدنه حلقه
 }
```

مثال: محاسبه معدل نمرات یک کلاس با هر تعداد دانش آموز

```
float sum = 0, aver;
Console.Write("Enter Numbers of Students:");
int m = int.Parse(Console.ReadLine());
for (int x = 1; x <= m;x++ )
{
    Console.Write("Enter Mark {0}:", x);
    float n=float.Parse(Console.ReadLine());
    sum = sum + n;//sum+=n;
```

```
}
aver = sum / m;
Console.WriteLine("Average is {0:f}", aver);
```

ب- دستور while : برای مواقعی که یک یا چند دستور باید تا زمانی که حاصل یک عبارت منطقی درست است اجرا شوند، از دستور while استفاده می‌شود.

While (عبارت منطقی)
{
دستورات بدنه حلقه
}

مثال: نمایش اعداد ۱۰ تا ۱ بصورت کاهشی

```
Int number=10;
while (number > 0)
{
Console. WriteLine(number);
number--;
}
```

مثال: محاسبه مجموع ارقام هر عدد دلخواه

```
int d,m,s=0;
Console.Write("Enter a Number:");
m = int.Parse(Console.ReadLine());
while( m > 0 )
{
    d=m%10;
    s += d;
    m = m / 10;
}
Console.WriteLine("number digits:{0}",s);
```

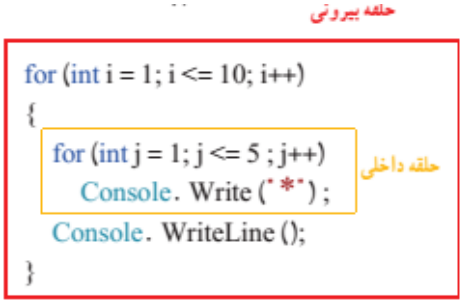
ج- دستور do-while

Do
{
دستورات بدنه حلقه
} while (شرط حلقه)

نکته: اگر در همان ابتدای اجرای حلقه های (while, do-while) شرط حلقه نادرست باشد حلقه while اصلا اجرا نمی شود ولی حلقه do-while حداقل یکبار اجرا می شود چون شرط حلقه در انتهای حلقه بررسی می شود.

حلقه های متداخل (تودرتو)

در بعضی از مواقع، یک دستور تکرار را در داخل دستور تکرار دیگری به کار می‌بریم. به عبارت دیگر هنگامیکه در داخل یک حلقه، حلقه دیگری قرار داشته باشد، حلقه‌های تودرتو یا متداخل نامیده می‌شوند



مثال: تولید جدول ضرب اعداد

```
int i,j;
for (i = 1; i <= 10; i++)
{
    for (j = 1; j <= 10; j++)
        Console.Write(“{0:4}”,i*j);
    Console.WriteLine();

}
```

مثال: خروجی زیر را تولید کنید.

```
int i,j;
for (i = 1; i <= 5; i++)
{
    for (j = 1; j <= i; j++)
        Console.Write("*");
    Console.WriteLine();

}
Console.ReadKey();
```

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

مثال: قطعه کد داده شده را trace کنید و خروجی را در جدول نمایش دهید.

```
for (int i=1; i<=4; i++)
{
      for (int j=4; j>=1; j--)
          Console.WriteLine((i==j)? 1:0);
      Console.WriteLine();
}
```

0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

کاراکتر \ (دنباله معنی دار یا دنباله فرار)

کاراکتر \ یک کاراکتر خاص و معنی دار است و هرگاه در داخل یک رشته دیده شود روی کاراکتر بعدی خود اثر گذاشته و سبب تغییر عملکرد کاراکتر بعدی می شود.

عملکرد	دنباله
ایجاد یک بوق هشدار[۲]	\a
حذف یک کاراکتر(Backspace)	\b
ایجاد یک خط خالی(New Line)	\n
ایجاد یک فاصله افقی زیاد tab	\t
ایجاد یک تک کوتیشن (')	\'
ایجاد یک دابل کوتیشن (")	\"
ایجاد یک Back slash (\)	\\

فصل دوم: آرایه (array)

- مکان های متوالی در حافظه کامپیوتر که در آن داده هایی از یک نوع نگهداری می شود آرایه نام دارد.

17	14	8	13	20

- به هر یک از این مکان ها یک عنصر آرایه می گویند
- به تعداد کل عناصر آرایه، طول آرایه گفته می شود.
- برای دسترسی به عناصر آرایه از اندیس استفاده می شود. اندیس آرایه از صفر شروع می شود.
- در برنامه هایی که بخواهیم از داده های ورودی چندین بار استفاده کنیم باید از آرایه کمک بگیریم.

نحوه‌ی تعریف آرایه

الف)تعریف آرایه بدون مقدار اولیه

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

مثال: تعریف آرایه از نوع صحیح و بطول ۵

```
Int [] list=new int[5];
```

ب)تعریف آرایه زمانیکه مقادیر آرایه از قبل مشخص است.

نوع داده [] نام آرایه =new نوع داده []{مقادیر آرایه};

یا بطور خلاصه:

نوع داده [] نام آرایه = {مقادیر آرایه};

مثال: تعریف آرایه ی رشته ای برای نگهداری نام روزهای هفته

```
String[] weekday={"Saturday","Sunday","Monday",...}
```

در این حالت طول آرایه برابر است با تعداد مقادیر(در این مثال طول آرایه برابر است با ۷)

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

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 برنامه نویس اشتباه کند و اندیس بالاتری را استفاده کند در هنگام ترجمه‌ی برنامه با خطا روبه رو می‌شود.

دسترسی به عناصر آرایه

نام آرایه [اندیس]

مثلا list[0] به عنصر اول آرایه و list[4] به عنصر آخر آرایه اشاره می کند.

مقدار دهی عناصر آرایه

الف- با دستور انتساب

```
List[0]=10;                مقدار دهی عنصر اول آرایه
```

ب- با استفاده از دستورات ورودی

```
List[0]=int.parse(console.readline());        مقدار دهی عنصر اول آرایه با داده ی ورودی
```

ج- استفاده از حلقه برای زمانی که تعداد عناصر زیاد است.

```
For(int i=0;i<5;i++)
  List[i]=int.parse(console.readline());        مقدار دهی کل عناصر آرایه
```

طول آرایه

با استفاده از ویژگی Length می توان طول آرایه را بدست آورد.

مثال: قطعه کد زیر طول آرایه را نمایش دهید.

```
int [] list={12,8,11,17,20};
console.writeline(list.Length);
```

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. دستورforeach روی هر عنصر از داده قابل شمارش تکرار می‌شود. مثلاً هر دو نوع array و string قابل شمارش هستند.

```
Foreach (نوع داده  نام متغیر  in  نام آرایه)
        دستور;
```

متغیر بکار رفته در این دستور با کلمه کلیدی in به آرایه متصل می شود و در هر بار اجرای حلقه نقش یک عنصر آرایه را به عهده دارد. به این ترتیب نیاز به کار با اندیس های آرایه برطرف می شود.

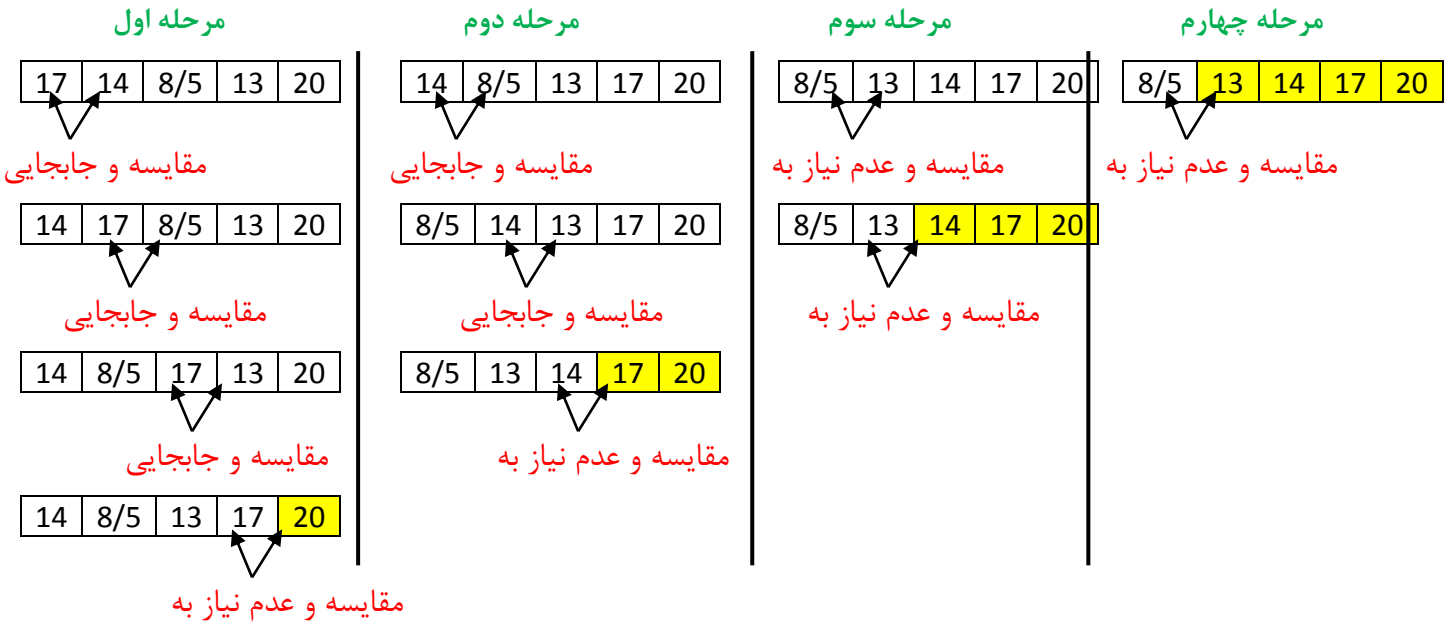
مثال: نمایش عناصر آرایه با حلقه foreach

```
int [] list={12,8,11,17,20};
foreach( int  x  in  list)
        console.writeline(x);
```

مرتب کردن داده های لیست (آرایه)

الف- مرتب سازی حبابی (Bubble Sort):

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 جا به جا می‌کنیم. سپس به سراغ عنصر دوم و سوم میرویم و عمل مقایسه و در صورت لزوم جابه جایی را انجام می دهیم. و تا آخر لیست به همین ترتیب جلو می‌رویم. به انتهای لیست که برسیم عنصر بزرگتر به انتهای لیست (جای اصلی خود) منتقل می‌شود. عمل مقایسه و جابجایی را مجدد تکرار می کنیم ولی این دفعه تا یکی مانده به آخر.(طبق رول زیر)



کد مربوط به مرتب سازی حبابی

```
float[] list = {17,14,8.5f,13,20 };
for(int i=list.Length-1;i>0;i--)
    for(int j=0;j<i;j++)
        if (list[j] > list[j + 1])
        {
            float temp = list[j];
            list[j] = list[j + 1];
            list[j + 1] = temp;
        }
foreach (float x in list)
    Console.WriteLine(x);
Console.ReadKey();
```

نکته: قطعه کد فوق، لیست را بصورت صعودی مرتب می‌کند. برای مرتب سازی نزولی کافی است علامت < به > تغییر کند.

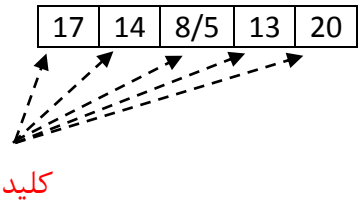
نکته: درالگوریتم مرتب سازی همواره دو عمل «مقایسه» و «جابه جایی »انجام می‌شود.

ب- مرتب سازی سریع (Quick Sort): مطالعه آزاد

جستجو در لیست

تعریف کلید جستجو(key): داده ای که به دنبال آن هستیم کلید جستجو نام دارد.

الف- جستجوی خطی (ترتیبی): در این روش کلید با تک تک عناصر موجود در لیست مقایسه می شود



کاربرد جستجوی خطی: در آرایه های کوچک و نامرتب استفاده می شود.

کد مربوط به جستجو در آرایه به روش خطی:

```
float[] list = {17,14,8.5f,13,20 };
float key = 13;
bool found = false;
int position = -1;
for (int i = 0; i < list.Length - 1; i++)
        if (list[i] == key)
        {
            found = true;
            position = i + 1;
            break;
        }
if(found)
    Console.WriteLine(position);
else
    Console.WriteLine("Not Found.");
```

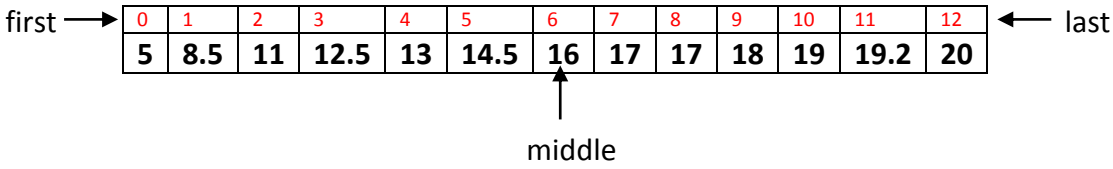
ب- جستجوی دودویی (باینری): در این روش کلید(key) با عنصر وسط(middle) لیست مقایسه می شود که سه حالت پیش می‌آید:

۱. Middle=key: پایان جستجو

۲. Middle<key: ادامه جستجو در نیمه راست (first=middle+1)

۳. Middle>key: ادامه جستجو در نیمه چپ (last=middle-1)

محاسبه اندیس وسط لیست = (first + last) / 2;



نکته۱: این جستجو بر روی لیست‌های مرتب کاربرد دارد. در صورتی که آرایه مورد نظر نامرتب باشد، ابتدا باید آن را مرتب کرد

نکته۲: با یک عمل مقایسه، منطقه مورد جستجو به اندازه نصف، کوچک می شود. 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

سوال: در یک آرایه با ۵۰۰ داده، حداکثر با چند مقایسه نتیجه جستجو مشخص می شود؟ جواب: ۹ مقایسه

مرحله	تعداد مقایسه
۱	۵۰۰
۲	۲۵۰
۳	۱۲۵
۴	۶۲
۵	۳۱
۶	۱۵
۷	۷
۸	۳
۹	۱

کد مربوط به جستجو به روش دودویی

```
float[] list = {17,14,8.5f,13,20 };
float key = 13;
bool found = false;
int first = 0;
int last = list.Length - 1;
int middle=(first + last) / 2;
while (last >= first)
{
    middle = (first + last) / 2;
    if (key == list[middle])
    {
        found = true;
        break;
    }
    else if (key > list[middle])
        first = middle + 1;
```

```
        else
            last = middle - 1;
    }
    if(found)
        Console.WriteLine(middle+1);
    else
        Console.WriteLine("Not Found.");
    Console.ReadKey();
```

فصل سوم: داده شمارشی، کلاس و متد

داده‌ی شمارشی

مجموعه ای از چند نام دلخواه که حالت‌ها و مقادیر مختلف یک موضوع را نشان می‌دهد، نوع داده شمارشی نام دارد. مثلاً برای روزهای هفته به جای اعداد 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.

نحوه‌ی تعریف نوع داده شمارشی

```
نام داده شمارشی  enum  نوع دسترسی
{
 لیستی از نام ها و کلمات (حالات مختلف داده شمارشی)
}
```

- روش نوشتن نام داده شمارشی مطابق با روش پاسکال است.
- محل قرارگیری تعریف نوع داده شمارشی، معمولاً خارج از کلاس و در ابتدای برنامه است.
- نوع دسترسی بطور پیش فرض public در نظر گرفته می شود.
- 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.
- 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.
- اگر مایل باشید می‌توانید عدد دیگری را برای نام‌ها اختصاص دهید.

مثال ۱: تعریف نوع داده شمارشی برای نگهداری فصل‌های سال

```
enum Season
{
        spring,
        summer,
        fall,
        winter
 }
```

در این تعریف :

۱. نام داده شمارشی Season در نظر گرفته شده است.
۲. نوع دسترسی ذکر نشده است بنابراین بطور پیش فرض public در نظر گرفته می شود.
۳. spring معادل عدد صفر است و summer معادل عدد یک و به همین ترتیب ادامه می یابد.
۴. هر یک از مقادیر داده شمارشی با کاما از هم مجزا شده اند.
۵. داده شمارشی باید خارج از کلاس تعریف شود.

مثال۲: تعریف نوع داده شمارشی برای نگهداری نام روزهای هفته

```
public   enum   DayOfWeek
{
        saturday=1,
        sunday,
        monday,
        tuesday,
        wednesday,
        thursday,
        friday
}
```

۱. در این تعریف اعداد معادل با داده شمارشی تغییر کرده است و حالت اول (Sunday) از یک شروع شده است.
۲. سوال: Friday معادل با چه عددی است؟ جواب : ۷

دسترسی به اعضای داده شمارشی:

نام حالت . نام داده شمارشی

مثال: دستور زیر اولین عضو داده شمارشی Season را نمایش می دهد.

```
Console.writeline(Season.spring);
```

تعریف متغیر از نوع داده شمارشی

نام متغیر نام داده شمارشی;

مثال: متغیر x از نوع داده شمارشی Season تعریف شده است و می تواند مقادیر spring, summer, fall, winter را بپذیرد.

```
Season  x;
```

مقداردهی متغیرهای شمارشی

نام حالت . نام داده شمارشی = نام متغیر

مثال: دومین عضو داده شمارشی Season در متغیر x قرار گرفته است.

```
x = Season.summer;
```

نکته ۱: نام گذاری متغیرها به روش کوهان شتری انجام می شود.

نکته ۲: مقداری که در متغیر نوع شمارشی قرار می‌گیرد باید با نوع آن مطابقت داشته باشد. مثلا در مثال بالا عدد ۱ (معادل summer) نمی تواند مستقیماً در متغیر x قرار بگیرد و باید از تبدیل نوع استفاده شود.

استفاده از نوع داده‌های شمارشی آماده در کتابخانه .NET.

ConsolColor یک نوع داده شمارشی آماده است که حالات مختلف رنگ در آن تعریف شده است.

برنامه نویسی شی گرایی(oop)

- 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 شیءگرا نامیده می شود.
- زبانهای C++ ، جاوا و C# از جمله زبانهای شئ گرا هستند.
- یک شئ شامل تعدادی ویژگی، وضعیت، رفتار و عملیات است که آن را از اشیای دیگر متمایز می‌سازد.
- ویژگیها و وضعیت یک شئ به وسیله تعدادی متغیر که فیلد نامیده می‌شوند، مشخص می‌شود.
- رفتارهای اشیاء در قالب متدها تعریف می‌گردند.

کلاس (class) : محل و مکان تعریف فیلدها و متدهای یک شئ

```
Class    نام شی
{
    ویژگی های شی (فیلدها)
    عملیاتی که شی انجام می
    دهد (متدها)
}
```

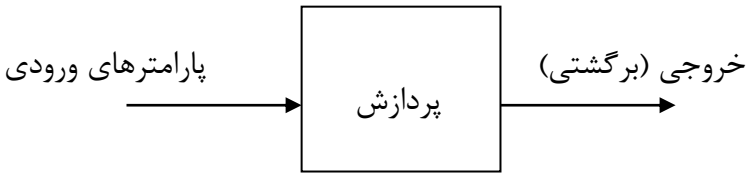
متد (method) : مجموعه ای از دستورات برای انجام یک عمل خاص متد نام دارد.

شکل تعریف متد:

```
(پارامترهای ورودی) نام متد  نوع داده خروجی  توصیف کننده
{
```

بدنه متد(دستورات)

}



توصیف کننده: دو تا چیز را مشخص می کند:

۱. محدوده ی دسترسی به متد (private, public)
۲. نحوه ی ایجاد متد (static)

نوع داده خروجی: یکی از انواع داده هاست و مربوط به داده ای است که متد آنرا محاسبه کرده و برگشت می دهد.

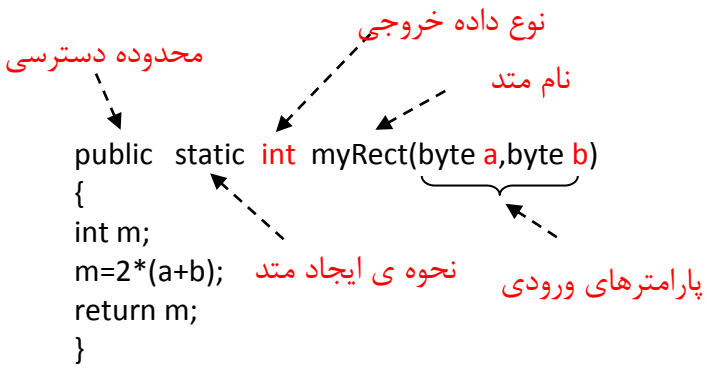
نکته: یک متد ممکن است خروجی نداشته باشد که در این صورت از void استفاده می کنیم.

نام متد: هر متد دارای یک نام است که برای اجرای آن استفاده می شود.

پارامترهای ورودی: چیزی که از بیرون به متد ارسال می شود.

نکته: ممکن است یک متد ورودی نداشته باشد که دراین صورت پرانتزهای خالی اجباری است.

مثال: متد زیر محیط یک مستطیل با طول و عرض a و b را محاسبه و نتیجه را برگشت می دهد.



نحوه ی اجرای متد (فراخوانی متد)

(لیست پارامترها)نام متد.کلاس.فضای نام

فصل چهارم: کار با متدها و کلاس های آماده

کلاس math

- زبان C# از طریق کتابخانه Net. انواع توابع ریاضی، مثلثاتی ولگاریتمی را برای برنامه‌نویس فراهم نموده است. این توابع به صورت متدهای استاتیک در کلاسMath تعریف شده‌اند.
- کلاس Math در فضای نامی System تعریف شده است و برای استفاده از متدهای آن باید نام System قبل از نام کلاس نیز ذکر گردد و یا در ابتدای برنامه با دستور using حوزه System به برنامه معرفی شود.

	نام	شکل استفاده	کاربرد	مثال	معادل ریاضی
متد	pow()	Math.pow(عدد,توان)	عدد را به توان مورد نظر می‌رساند.	Math.pow(n,3)	n^3
	sqrt()	Math.Sqrt(عدد)	محاسبه‌ی جذر (رادیکال) عدد	Math.sqrt(x)	$\sqrt{x}$
	truncate()	Math.Truncate(عدد)	بخش اعشار عدد را حذف می‌کند	Math.truncate(x)	$[x]$
	round()	Math.round(عدد)	عدد را گرد می‌کند	Math.round(x)	-
	max()	Math.max(عدد۱,عدد۲)	پیدا کردن عدد بزرگتر	Math.max(x,y)	-
	min()	Math.min(عدد۱,عدد۲)	پیدا کردن عدد کوچکتر	Math.min(x,y)	-
	log()	Math.log(عدد,مبنا)	محاسبه‌ی لگاریتم عدد در مبنای داده شده	Math.log(n,2)	$\log_2 n$
	Abs()	Math.abs(عدد)	محاسبه‌ی قدرمطلق یک عدد	Msth.abs(n)	$\|n\|$
	sin()	Math.sin(عدد)	سینوس یک زاویه (بر حسب رادیان) را حساب می‌کند	Math.sin(30)	$\sin 30$
	cos()	Math.sin(عدد)	کسینوس یک زاویه (بر حسب رادیان) را حساب می‌کند	Math.cos(30)	$\cos 30$
	tan()	Math.sin(عدد)	تانژانت یک زاویه (بر حسب رادیان) را حساب می‌کند	Math.tan(30)	$\tan 30$
ثابت	PI	Math.PI	عدد π را برمیگرداند.	Math.PI	π

توجه داشته باشید که: PI متد نیست بلکه یک ثابت است.(جلوی آن نباید از پرانتز استفاده شود)

کلاس string

- نوع داده string در واقع یک کلاس است و متغیری که از این نوع تعریف می‌شود اشاره به ردیفی از کاراکترها می‌کند.
- برای آنکه به حروف یک رشته دسترسی داشته باشیم می‌توانیم مانند آرایه‌ها، از اندیس حروف استفاده کنیم.
- پس از اینکه یک رشته در حافظه ایجاد شد، محتویات آن قابل تغییر نیست. (تغییر ناپذیری رشته)
- اگر متغیری را از نوع string تعریف نماییم، این متغیر دارای متدهایی خواهد بود که لیست آنها در جدول زیر آمده است.

	نام	کاربرد	شکل استفاده (برای رشته‌ی s)	مثال (s="visual studio 2012")	نتیجه
متد	CompareTo()	مقایسه دو رشته و برگشت یکی از مقادیر ۰ و ۱ و ۱-	s.CompareTo("رشته")	s.CompareTo("visual studio 2010");	1
	IndesOf()	پیدا کردن موقعیت یک کاراکتر در رشته و برگشت اندیس	s.IndexOf('کاراکتر')	s.IndexOf('u');	3
	ToUpper()	تبدیل تمامی کاراکترهای رشته به بزرگ	s.ToUpper()	s.ToUpper();	VISUAL STUDIO 2012
	ToLower()	تبدیل تمامی کاراکترهای رشته به کوچک	s.ToLower()	s.ToLower();	visual studio 2012
	Insert()	درج یک کاراکتر یا رشته در رشته ی دیگر	s.Insert(موقعیت , رشته)	s.Insert(14,"express");	visual studio express 2012
	Replace()	جایگزین کردن یک کاراکتر یا رشته با کاراکتر یا رشته دیگر	s.Replace(رشته موجود , رشته جایگزین)	s.Replace("2012","express");	visual studio express
ویژگی	Length	بدست آوردن طول رشته (تعداد کاراکتر)	s.Length	s.Length	18

نکته: این متدها روی خود رشته تأثیر نمی‌گذارند بلکه یک رشته دیگر بر می‌گردانند. بنابراین محتوای متغیر رشته‌ای را تغییر نمی‌دهند.

کلاس array

از ویژگی‌های این کلاس، وجود متدهای مختلف استاتیک، برای عملیات بر روی آرایه‌ها است.

	نام	کاربرد	شکل استفاده	مثال برای آرایه های int [] list={11,5,16,19,13,16,20} int [] grade=new int[5];	نتیجه
متد	Sort()	مرتب کردن عناصر آرایه (صعودی)	Array.Sort(نام آرایه);	Array.Sort(list);	List={5,11,13,16,16,19,20}
	Reverse()	وارونه کردن عناصر آرایه	Array.Reverse(نام آرایه);	Array.Reverse();	List={20,16,13,19,16,5,11}
	IndexOf()	جستجوی یک مقدار و برگرداندن اولین مورد پیدا شده	Array.IndexOf(نام آرایه , مقدار);	Array.IndexOf(list,19);	3
	LastIndexOf()	جستجوی یک مقدار و برگرداندن آخرین مورد پیدا شده	Array.LastIndexOf(نام آرایه , مقدار);	Array.LastIndexOf(list,16);	5
	BineraySearch()	جستجوی یک مقدار در آرایه مرتب (صعودی) و برگرداندن اولین مورد پیدا شده	Array.BinarySearch(نام آرایه , مقدار);	Array.BinarySearch(list,13);	4
	Copy()	کپی کردن عناصر یک آرایه در آرایه دیگر از اولین عنصر تا تعداد تعیین شده	Array.Copy(آرایه مبدا, آرایه مقصد , تعداد);	Array.Copy(list,grade,3);	Grade={11,5,16,0,0}
	Clear()	پاک کردن و تنظیم مقدار عناصر تعیین شده آرایه با مقدار پیش فرض نوع داده آرایه	Array.Clear(نام آرایه , اندیس شروع , تعداد);	Array.Clear(list,0,3);	List={0,0,0,19,13,16,20}
ویژگی	Length	بدست آوردن طول آرایه (تعداد عناصر آرایه)	نام آرایه.Length	List.Length;	7

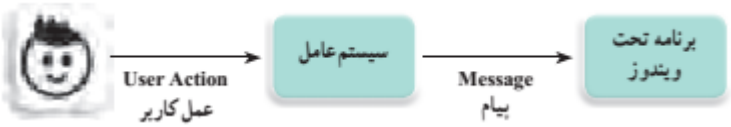
فصل پنجم : ایجاد برنامه های ویندوزی با Visual Studio

تفاوت برنامه‌های کنسولی و برنامه‌های ویندوزی

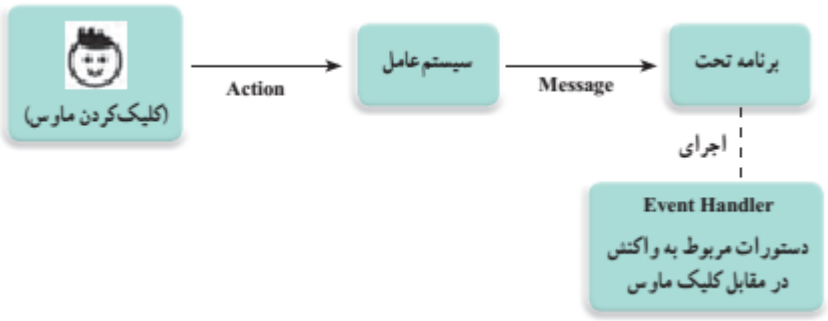
در برنامه‌های کنسولی، اجرای برنامه از متدی به نام Main() شروع می‌شود و دستورات داخل آن، به ترتیب و خط به خط اجرا می‌شوند. در چنین برنامه‌هایی با استفاده از متدهای موجود مانند ReadLine() برای دریافت یک رشته، درخواست‌هایی به سیستم عامل داده می‌شود که عملیاتی را برای ما انجام دهد.

در برنامه‌های ویندوزی نیز اجرای برنامه با متدی به نام Main() شروع می‌شود اما برخلاف برنامه‌های کنسول، برنامه در حالت انتظار قرار می‌گیرد تا یک اتفاق یا رویداد رخ دهد که در این صورت نسبت به آن واکنش نشان دهد.

رویداد: یک اطلاع است که از طرف سیستم عامل به برنامه داده می‌شود تا نشان دهد که یک اتفاق رخ داده است.



بنابراین در برنامه‌های ویندوزی، باید پیش بینی کنید که اگر کاربر عملی را انجام دهد، برنامه چگونه نسبت به آن واکنش نشان دهد و برای این منظور باید متدهایی را بنویسید که در مواجهه با یک رخداد یا رویداد مانند کلیک ماوس در یک محل، به کامپیوتر اعلام کند که چه کاری باید انجام شود. به این متدها، Event Handler(EH) گفته می‌شود .



شکل کلی یک متد EH به صورت زیر است:

```
Void   نام متد (جزئیات رویداد , فرستنده پیام)
{
        دستورات واکنش نسبت به رویداد
}
```

یک تفاوت دیگر بین برنامه ویندوزی و کنسولی این است که، برنامه‌های ویندوزی پس از واکنش به رویدادها و انجام عملیات مربوطه، مجددا در حالت انتظار برای رویداد بعدی به سر می‌برند تا در نهایت کاربر از برنامه خارج گردد.

در ضمن با توجه به اینکه رویدادهای مختلفی ممکن است از طریق کاربر رخ دهد که قابل پیش‌بینی نیست، بنابراین با اجرای برنامه، مشخص نیست که کدام رویداد ممکن است ابتدا رخ دهد و به عبارت دیگر ترتیبی برای رویدادها، قابل پیش بینی نیست.

واسط گرافیکی کاربر (GUI)

در ساخت یک برنامه ویندوزی باید صفحه یا فرمی در اختیار داشته باشید تا انواع دکمه‌ها، منوها، تصاویر و نوشته‌ها را روی آن قرار دهید. این فرم، واسط گرافیکی کاربر نامیده می شود.

آشنایی با پنجره‌های IDE

الف) منوها و نوار ابزارها: در محیط ویندوز، یک منو به نام Format اضافه شده است که از گزینه‌های آن برای شکل‌دهی ظاهری و تنظیم جای دهی کنترل‌های یک فرم استفاده می‌شود.

ب) جعبه ابزار(toolbox): در پنجره جعبه ابزار، نام و آیکن کنترل‌ها بصورت دسته بندی، لیست شده است. برای قرار دادن کنترل بر روی فرم می‌توان:

- روی کنترل دابل کلیک کرد
- کنترل را به روی فرم درگ کرد

ج) فرم(form): در وسط صفحه، یک فرم دیده می‌شود که از آن برای طراحی واسط کاربری و قراردادن کنترلها استفاده می‌شود. فرم در دو نما قابل مشاهده است:

- حالت طراحی(designer)
- حالت code

د) پنجره solution explorer: در این پنجره، دستورات یک برنامه ویندوزی در سه فایل قابل مشاهده است:

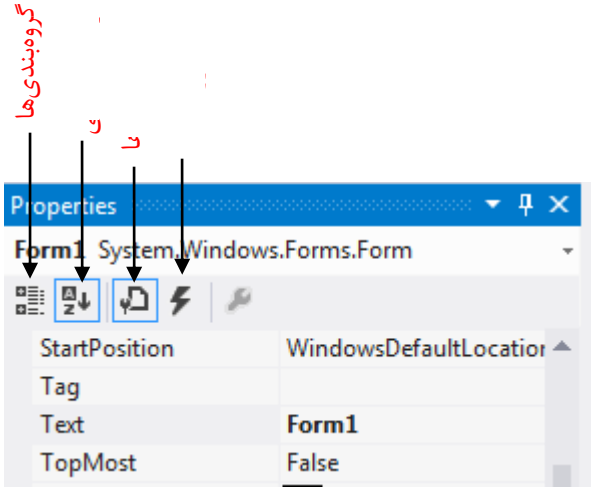
- فایل program.cs: با دابل کلیک روی این فایل محتویات آن قابل مشاهده است. کلاس اصلی برنامه و متد Main() در این فایل قرار دارد.
- فایل form1.cs: با کلیک راست روی این فایل و انتخاب view code محتوای آن که قسمتی از تعریف کلاس Form1 است نشان داده می شود. کلمه کلیدی partial در خط عنوان کلاس form1 نشان می‌دهد که بخشی از تعریف کلاس Form1 در فایل دیگری قرار دارد.
- فایل Form1.Designer.cs: تنظیمات مربوط به کنترل های روی فرم به صورت دستورات زبان C# بعنوان بخشی از تعریف کلاس form1 در این فایل نگهداری می‌شود.

اگر در پنجره Solution Explorer بر روی فایل Form1.cs کلیک راست کنید، منویی ظاهر می‌شود که شامل گزینه‌هایی برای انجام عملیات مختلف بر روی آن فایل است. به این منوها که گزینه‌های آن مربوط به یک موضوع خاص است منوی موضوعی یا محلی گفته می‌شود. از طریق این منو می‌توان فرم را در نمای کد یا طراحی مشاهده کرد.

همچنین می‌توان از کلیدهای میانبر نیز این کار را انجام داد:

- مشاهده فرم در نمای طراحی: shift+f7
- مشاهده فرم در نمای کد (دستورات فایل فرم): alt+ctrl+0

ه) پنجره ویژگیها و خواص اشیاء (properties)



- در این پنجره ویژگیها و خواص شئ (properties) و همچنین واکنش به رویدادهای هر شئ (events) را می‌توانیم مقداردهی و تنظیم کنیم.
- در قسمت بالای این پنجره، نام یک شئ که در فرم انتخاب شده، نشان داده شده است و از طریق مثلث کوچک کنار آن میتوان شئ دیگری را از بین اشیای موجود روی فرم

انتخاب کنیم.

- لیست ویژگی‌ها و رویدادها به تریب نام (الفبایی) و یا به ترتیب عملکرد(categories) قابل مشاهده هستند.

نکته۱: پنجره‌ها در حالت Auto Hide قرار دارند یعنی با کلیک بر روی نام آنها، پنجره دیده می‌شود و با کلیک در نقطه دیگری از صفحه، پنجره‌ها به طور خودکار کنار می‌روند و فقط نام پنجره‌ها دیده می‌شوند. از قابلیت Auto Hide برای کنار رفتن موقتی پنجره‌ها و برای بزرگ‌شدن محیط کار استفاده می‌شود. برای اینکه پنجره‌ای را از حالت Auto Hide خارج کنید و یا این قابلیت را به آن بدهید، کافی است در روی خط عنوان پنجره بر روی آیکن پونز کلیک کنید.

۱. آیکن پونز در حالت افقی: قابلیت auto hide فعال است.
۲. آیکن پونز در حالت عمودی: قابلیت auto hide غیر فعال است.

در این حالت پنجره همیشه روی صفحه ثابت و قابل مشاهده است.

نکته۲: پنجره‌های را می‌توان درگ کرد و آنها را به صورت شناور(float) تبدیل کرد.

نکته۳: پنجره‌ها را می‌توان به صورت tab docked تنظیم کرد تا از یک فضا بصورت مشترک استفاده کنند.

نکته۴: برای تنظیم پنجره ها به صورت float یا tab docked از مثلث کنار آیکن پونز استفاده می‌شود.

نکته۵: اگر پنجره‌ها قابل مشاهده نیستند می‌توان آنها را از منوی window فعال کرد.

آشنایی با خواص و ویژگی‌های فرم

ویژگی	عملکرد
Name	نامگذاری فرم
Text	تعیین خط عنوان فرم
RightToLeft	پشتیبانی از زبانهای راست به چپ (مانند فارسی)
RightToLeftLayout	طرح بندی فرم به صورت راست به چپ
ControlBox	وجود یا عدم وجود دکمه های بستن، بیشینه و کمینه
FormBorderStyle	اندازه فرم متغیر(sizable) یا ثابت(fixed)
Size(width/height)	اندازه فرم (پهنا و ارتفاع)

نکته: اگر دکمه بستن فرم (ضربدر) غیر فعال شده باشد برای بستن فرم در زمان اجرا از کلیدهای Alt+F4 استفاده می‌شود.

آشنایی با کنترل برچسب(Label)

برای اضافه کردن یک متن نمایشی بر روی فرم بکار می‌رود.

ویژگی	عملکرد
Name	نام برچسب
Text	تعیین عبارت روی برچسب
RightToLeft	پشتیبانی از زبانهای راست به چپ (مانند فارسی)
Font	فونت یا نوع قلم

ForeColor	تعیین رنگ نوشته
Location	موقعیت برچسب روی فرم که بر اساس فاصله از چپ(x) و بالای(y) فرم سنجیده می شود.
Visible	وضعیت دیده شدن یا مخفی بودن کنترل (کنترل قابل مشاهده باشد یا نه). مقدار true باعث نمایش کنترل و مقدار false باعث مخفی شده کنترل می‌شود.
BackColor	رنگ زمینه

آشنایی با کنترل جعبه تصویر (PictureBox)

برای اضافه کردن عکس بر روی فرم بکار می‌رود.

ویژگی	عملکرد
Name	نام کنترل جعبه تصویر
Image	تعیین تصویر (با استفاده از مثلث اطراف کادر تصویر نیز می‌توان این کار را انجام داد)
SizeMode	نوع تعیین اندازه تصویر: • StretchImage: اندازه تصویر با اندازه کنترل هماهنگ می‌شود. • AutoSize : اندازه کنترل با اندازه تصویر هماهنگ می‌شود.
Anchor	ثابت کردن فاصله کنترل نسبت به لبه‌های فرم
ImageLocation	آدرس یک تصویر را می‌گیرد و آن را در کادر تصویر بارگذاری می‌نماید.

پوشه منابع(Resources)

تصاویری که در پروژه استفاده می‌کنیم در محل ذخیره پروژه در پوشه‌ای بنام Resources نگهداری می‌شوند که به آن پوشه منابع می‌گویند.

فصل ششم : ایجاد برنامه های ویندوزی با واکنش نسبت به رویدادها

در این فصل علاوه بر آشنایی با کنترلهای جدید، با رویدادهای مربوط به چند کنترل آشنا خواهیم شد.

رویداد(event)

در برنامه‌های ویندوزی، باید پیش بینی کنید که در زمان اجرای برنامه اگر کاربر عملی را انجام دهد، برنامه چگونه نسبت به آن واکنش نشان دهد و برای این منظور باید متدهایی را بنویسید که در مواجهه با یک رخداد یا رویداد مانند کلیک ماوس روی دکمه، به کامپیوتر اعلام کند که چه کاری باید انجام شود. لیست این رویدادها را می‌توان در پنجره properties مشاهده نمایید.

آشنایی با کنترل دکمه (button)

	نام	عملکرد
ویژگی ها	Name	نام دکمه
	Text	متن روی دکمه
	Font	قلم
	AutoSize	وضعیت اندازه دکمه
	RightToLeft	پشتیبانی از زبانهای راست به چپ
	Location	محل دکمه
	BackgroundImage	تصویر زمینه. در این حالت نوشته ی روی دکمه قابل مشاهده است.
	BackgroundImageLayout	نوع نمایش تصویر زمینه که اگر بر روی Stretch تنظیم شود کل تصویر بر روی دکمه قابل مشاهده خواهد بود.
	Image	تصویر دکمه. در این حالت نوشته ی روی دکمه قابل مشاهده نیست.
رویداد	Click()	در زمان اجرای برنامه اگر کاربر روی دکمه کلیک کند این رویداد رخ می‌دهد و دستورات نوشته شده در این رویداد اجرا خواهند شد. در زمان طراحی فرم با دابل کلیک روی دکمه می‌توان دستورات مورد نظر خود را در این رویداد وارد کرد.

نکته۱: علاوه بر نوشتن متن روی دکمه می‌توان از علامت یا نماد (symbol) نیز روی دکمه استفاده کرد. در این صورت باید قلم wingdings انتخاب شود و در قسمت ویژگی text کلید alt را گرفته و کد اسکی نماد را از طریق کلیدهای ماشین حسابی وارد کرد.

نکته۲: برای قرار دادن یک کنترل دقیقا در مرکز فرم می‌توان از منوی format فرمان center in form استفاده کرد.

نکته۳: ویژگی‌های کنترل‌ها را علاوه بر پنجره properties می‌توان از طریق کدنویسی نیز تغییر داد به مثال زیر دقت کنید.

مثال: می‌خواهیم با کلیک روی دکمه، برچسب مخفی موجود بر روی فرم با نام label1 نمایان شود، کد زیر را در رویداد کلیک دکمه وارد می‌کنیم:

```
Label1.visible=true;
```

نکته۴: اگر بخواهید از منابع معرفی شده در نرم افزار (تصویر و ...) خود در زمان اجرا (در کد) استفاده کنید باید از کلاس Resources که در فضای نام Properties است، به صورت زیر استفاده نمایید.

```
Properties. Resources. نام منبع
```

مثال: می‌خواهیم با کلیک روی دکمه، تصویر موجود در پوشه منابع (pic1) در کنترل جعبه تصویر (picturebox1) نمایش داده شود. کد زیر را در رویداد کلیک دکمه وارد می‌کنیم:

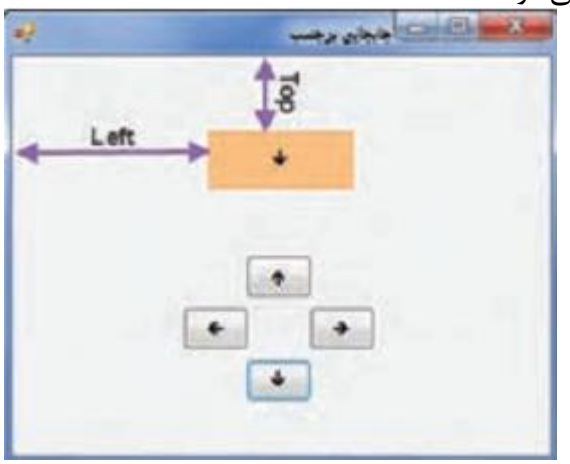
```
PictureBox1.Image= Properties.Resources.pic1;
```

مثال: می‌خواهیم با کلیک روی دکمه (Button1)، تصویر موجود در زمینه‌ی دکمه، در جعبه تصویر (PictureBox1) نمایش داده شود. کد زیر را در رویداد کلیک دکمه وارد می‌کنیم:

```
PictureBox1.Image=Button1. BackgroundImage;
```

جابجایی کنترل‌ها روی فرم در زمان اجرا

در زمان طراحی فرم، موقعیت هر کنترل نسبت به فرمی که روی آن قرار دارد با ویژگی Location که دارای دو مشخصه X و Y است تعیین می‌شود ولی برای تغییر موقعیت کنترل در زمان اجرا از دو ویژگی Left و Top که به ترتیب معادل X وY هستند استفاده می‌کنیم. بنابراین مشخصه Left فاصله کنترل از سمت چپ فرم و مشخصه Top فاصله کنترل از بالای فرم است بنابراین هر چه به مقدار مشخصه Left اضافه شود، فاصله کنترل از سمت چپ فرم بیشتر می‌شود و کنترل به سمت راست جابه‌جا می‌شود. هر چه از مقدار مشخصه Left کم شود، فاصله کنترل از سمت چپ فرم کمتر می‌شود و کنترل به سمت چپ جابه‌جا می‌شود. به همین ترتیب هر چه به مقدار مشخصه Top اضافه شود، فاصله کنترل از بالای فرم بیشتر می‌شود و کنترل به سمت پایین جابه‌جا می‌شود. هر چه از مقدار مشخصه Top کم شود، فاصله بالای کنترل از بالای فرم بیشتر می‌شود و کنترل به سمت بالا جابه‌جا می‌شود.



مثال: می خواهیم با کلیک روی دکمه، کنترل برچسب (Label1) به سمت راست حرکت کند. کد زیر را در رویداد کلیک دکمه وارد می‌کنیم:

```
Label1.Left+=1;
```

کنترل زمان سنج (timer)

کنترل زمان‌سنج برای اطلاع از سپری شدن فاصله‌های زمانی یکسان به کار می‌رود. برای مثال اگر با رایانه کار می‌کنید، برای حفظ سلامتی لازم است هر نیم ساعت چند نرمش مناسب انجام دهید. ساعت را تنظیم می‌کنید که هر نیم ساعت زنگ بزند. زنگ ساعت به شما اطلاع می‌دهد که زمان نرمش رسیده است. کنترل زمان‌سنج این کار را برای برنامه انجام می‌دهد.

	نام	عملکرد
ویژگی‌ها	Name	نام کنترل
	Interval	فاصله زمانی بر حسب میلی ثانیه. (هر ثانیه برابر است با ۱۰۰۰ میلی ثانیه)
	Enable	وضعیت فعال یا غیر فعال بود کنترل با مقادیر true و false

رویداد	Tick()	این رویداد در فواصل زمانی یکسان که در ویژگی Interval کنترل مشخص می‌شود رخ می‌دهد (مشابه زنگ ساعت)

نکته: این کنترل پایین فرم قرار می‌گیرد و در زمان اجرا دیده نمی‌شود

کنترل کادر محاوره ای انتخاب فایل (OpenFileDialog)

	نام	عملکرد
ویژگی ها	Name	نام کنترل
	Filter	تعین فرمت فایل‌های قابل انتخاب
	FileName	نگهداری نام و مسیر فایل انتخاب شده در کادر محاوره ای open
رویداد	ShowDialog()	باز کردن کادر محاوره ای
	FileOk()	با انتخاب فایل در کادر OpenFileDialog این رویداد رخ می‌دهد.

نکته۱: می‌توانید توسط ویژگی Filter نوع فایل‌هایی که کادر محاوره‌ای نشان می‌دهد را مشخص کنید. مقداردهی این ویژگی به صورت زیر است:

...|نوع۲ |شرح نوع۲ |نوع۱ |شرح نوع ۱

مثال : Jpg Files|*.jpg|Gif Files|*.Gif

نکته۲: انتخاب فایل با دابل کلیک روی نام فایل یا کلیک روی نام فایل و زدن دکمه open در کادر OpenFileDialog انجام می‌شود.

مثال: کد زیر تصویر انتخاب شده توسط کادر محاوره‌ای open را در جعبه تصویر قرار می‌دهد.

```
pictureBox1.ImageLocation = openFileDialog1.FileName;
```

کنترل کادر محاوره‌ای فونت (FontDialog)

برای تغییر قلم، اندازه و رنگ آن از کادر محاوره‌ای فونت استفاده می‌شود.

	نام	عملکرد
ویژگی ها	Name	نام کنترل
	ShowColor	امکان انتخاب رنگ را در کادر محاوره‌ای فونت فراهم می‌کند.
رویداد	ShowDialog()	باز کردن کادر محاوره ای فونت

کنترل کادر محاوره‌ای رنگ (ColorDialog)

برای تغییر رنگ از کادر محاوره‌ای رنگ استفاده می‌شود.

کنترل جعبه متن (TextBox)

اگر بخواهیم در زمان اجرای برنامه اطلاعاتی از کاربر دریافت کنیم از این کنترل استفاده می‌شود.

	نام	عملکرد
ویژگی ها	Name	نام کنترل
	ShowColor	امکان انتخاب رنگ را در کادر محاوره‌ای فونت فراهم می‌کند.
	MultiLine	جعبه متن چند خطی ایجاد می‌کند.
	ReadOnly	جعبه متن برای کاربر مانند کنترل برچسب می‌شود و کاربر نمی‌تواند چیزی در آن بنویسد.

مثال: می‌خواهیم فونت و رنگ نوشته جعبه متن را از طریق کادرهای محاوره ای قلم تغییر دهیم، در رویداد کلیک دکمه می‌نویسیم:

```
fontDialog.showDialog();
TextBox1.forecolor=fontDialog.color;
TextBox1.font=fontDialog.font;
```

کنترل کادر علامت (CheckBox)

برای دادن امکان انتخاب ویژگی‌ها به کاربر از کنترل CheckBox استفاده می‌کنیم. این کنترل در مواردی که دوحالت داریم مثل true/false یا بله / خیر کاربرد دارد. در کنار این کنترل مربع کوچکی قرار دارد که در زمان اجرا با کلیک روی آن می‌توان علامت در آن قرار داد و یا علامت آن را حذف کرد.

	نام	عملکرد
ویژگی ها	Name	نام کنترل
	Text	متن کنار کنترل
	checked	وضعیت انتخاب یا عدم انتخاب
رویداد	CheckedChanged()	در زمان اجرا با کلیک روی کادر علامت این رویداد رخ می‌دهد.