آموزش مقدماتی سی شارپ

سال تحصیلی 90-1389

تحقیق ARC

مدرسه ی راهنمایی دکتر محمود افشار

نویسنده: محمد تحویلداری کلاس 3/1

ایمیل: Mohammad.Tahvildary@gmail.com سایت: http://ShamotCity.vcp.ir

ناشر: گروه شاموت!

فهرست

موضوعات

جلسه یک
مقدمه

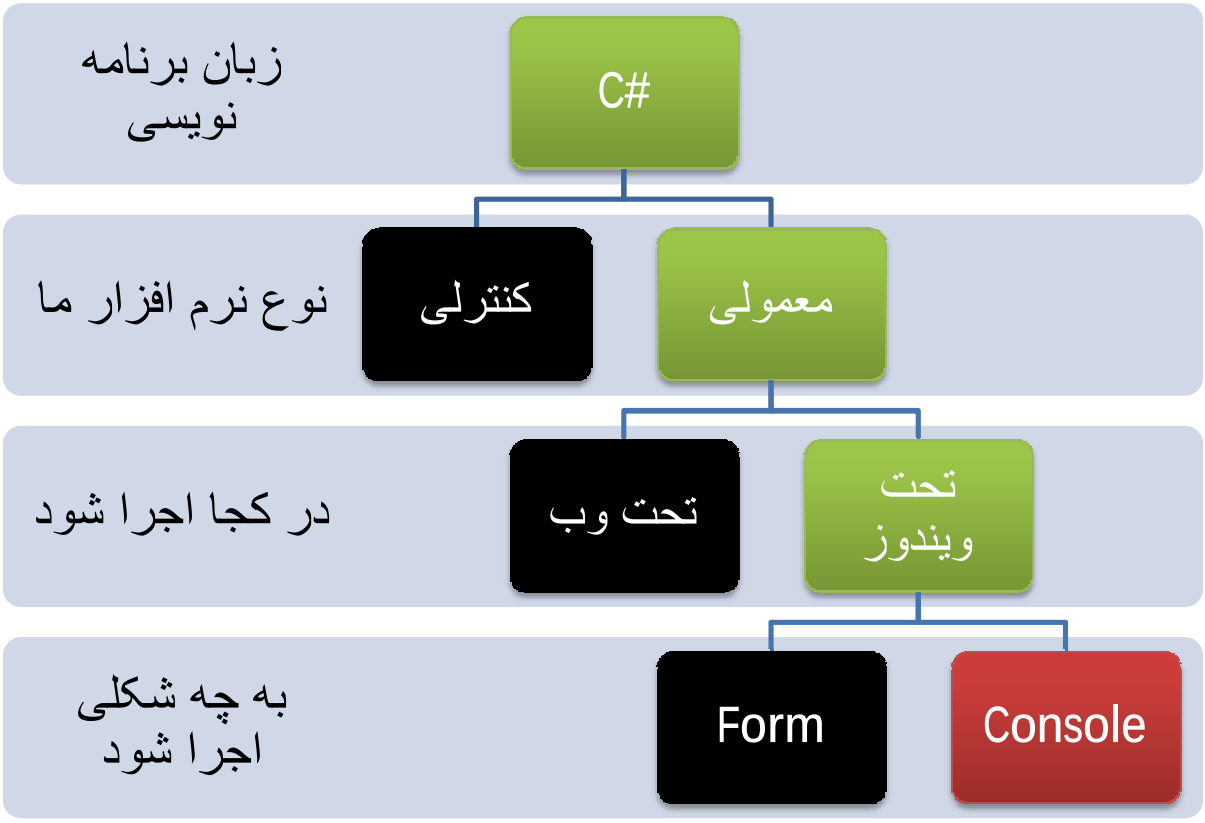
در این جلسه ما قصد معرفی کار هایی که شما در آینده انجام خواهید داد را داریم.

حال به امید خدا اولین جلسه را شروع می کنیم:

آیا تا به حال برنامه ای یا صفحه ی وبی و یا این چنین برنامه هایی را نوشته اید؟ اگر که پاسخ شما بله است که هیچ. ولی اگر من اولین کسی هستم که افتخار آموزش این درس ساده را دارم باید بهتان بگویم که از این پس شما دیگر حق ندارید که بگویید: من بلد نیستم!چون از حالا دارید یاد می گیرید. پس نباید ساختن برنامه هایی مثل پک ادوبی ویا مایکروسافت و حتی نوشتن یک سیستم عامل را دشوار بدانید. شاید برایتان جالب باشد که من هم بلد نیستم که سیستم عاملی طراحی کنم. اما امید دارم و همین امید است که باعث می شود آدمی بر پا بماند. خواهش می کنم دست به ماوس نزنید و صفحه را نبندید.این ها یک سری مطلب بود که بایستی بدانید.حال یک ذره بحث را تخصصی تر می کنیم.

شما ابتدا باید بدانید که ما قرار است با یکی از محبوب ترین زبان های دنیا(در تاریخ جمعه، 17/09/2010) برنامه های خود را که قرار است چیز های جالبی شوند را بنویسیم. در ضمن تنها ابزار مورد نیاز ما نرم افزار مایکروسافت ویژوال استودیو 2008()است. البته نسخه ی 2010 هم آمده ولی چون که با سیستم شما کامل مچ باشد و در همه جا پیدا شود این نسخه را استفاده می کنیم. البته من دروغ نمی گویم، دلیل دیگرش هم این است که من خودم با این نسخه سازگاری بیشتری دارم! نرم افزار ویژوال استودیو فقط از زبان سی شارپ پشتیبانی نمی کند. یعنی جز این که ما نرم افزار های خود را با سی شارپ می توانیم بنویسیم، با زبان هایی مثل VB، C++ و... هم می توانیم این کار را به خوبی انجام دهیم.

ما در سی شارپ دو دسته نرم افزار داریم. یکسری کنترلی است و کاربرنقشی در اجرای آن ها ندارند و اصولا در پس زمینه اجرا خواهند شد.یک سری دیگر هم نرم افزار های معمولی است. ما در این دوره بخشی از این نرم افزار ها را یاد می گیریم که بنویسیم.



جدول شاخه های سی شارپ

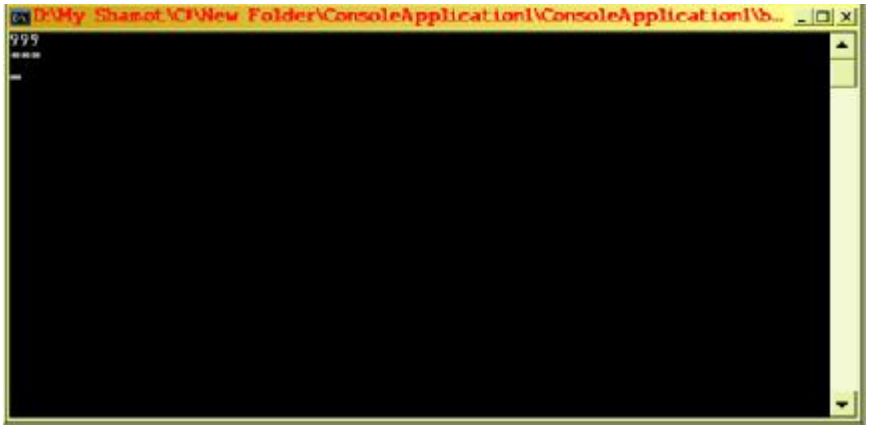
خب همانطور که مشاهده می کنید در جدول بالا مابرنامه های معمولی و کاربری(یعنی برنامه هایی مثل فوتوشاپ، اپرا، فایرفاکس و...) می نویسیم. یعنی برنامه هایی که کاربر در اجرای عملیات نقش دارد. بعد این دسته ی نرم افزار های معمولی دو رشته می شوند. این دو رشته تحت ویندوز و تحت وب می باشد که بعدا در دیگر دوره ها به آن پرداخته می شود. خب محل کنونی ما قسمت کنسول هست که در واقع بگذارید رک باشیم. این قسمت ساده ترین قسمت برنامه نویسی است. البته یک ضرب المثل هست که می گوید: گاماس گاماس. یعنی قدم به قدم. شما ابتدا باید اصول را یاد بگیرید

تا بتوانید یک موتور جستجوی قدرتمند تر از گوگل یا یاهو بسازید. ابتدا باید از بیس(Base) برنامه نویسی آموزش ببینید تا بتوانید بزرگ ترین شرکت کامپیوتری را در ایران ثبت کنید ویا بهترین سایت آپلود و با سرور های قدرتمند داشته باشید. ویا...

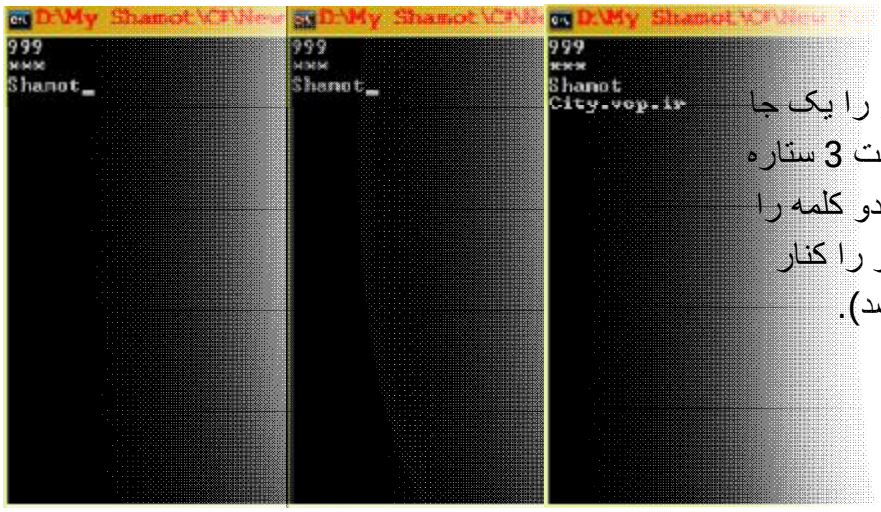
همان طور که گفته شد نرم افزار های معمولی دو نوع فرم و کنسول هستند.

کنسول(Console): به یک سری برنامه می گویند که در سیستم عامل لینوکس با نام گنو وجود دارد و در ویندوز هم نامش کنسول است! این نوع از نرم افزار ها به هیچ وجه گرافیکی نیستند و بیش تر کار هایشان محاسبه و یا شانس و آمار و... است.

مثال:



این یک نوع کنسول است.این برنامه قرار است فقط یک 999 را بنویسد. دقت کنید که این برنامه ای که الان نوشته شده است یک برنامه ی آموزشی و صرفا کسی برای فقط نوشتن یک عدد (999) برنامه نمی نویسد!



چون این برنامه چند چیز را یک جا انجام می دهد بین هرقسمت 3 ستاره موجود است. این قسمت دو کلمه را می گیرد و در آخر آن دو را کنار هم چاپ می کند(می نویسد).

این قسمت هم دو عدد می گیرد(یعنی کاربر وارد می کند)، جمع، ضرب، تفریق و تقسیم آن دو را حساب کرده و چاپ می کند.

خب بگذارید یک ذره از این بحث دور شویم و به نکاتی درباره ی ویژوال استودیو ی 2008 بپردازیم. موقع نصب این برنامه باید دقت کنید که این نرم افزار بسیار کامل است و شما احتیاجی ندارید که کل آن را نصب کنید. کافیست قسمت C# را نصب کنید. آن هم فعلا تحت ویندوز را. البته قسمت Help این برنامه خیلی کامل است و اگر از دانش انگلیسی تخصصی بهره مندید واقعا نصب کامل راهنمای برنامه توصیه می شود. بیایید رو راست باشیم. شاید خیلی کامل تر از این دوره هم باشد!

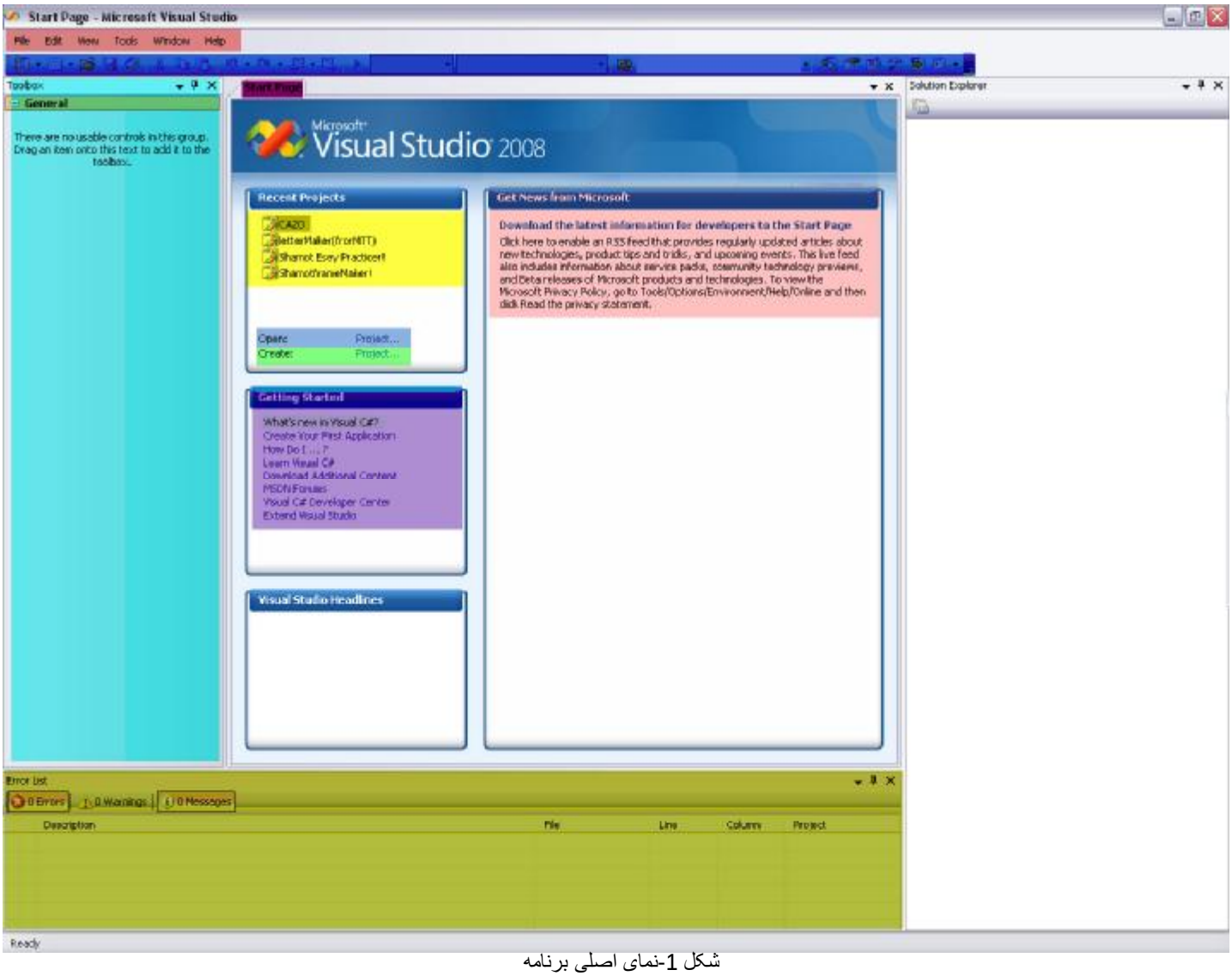
فکر کنم برای این جلسه کافی باشد.جلسه ی بعد مطالبی را راجع به طرز کار با مایکروسافت ویژوال استودیو 2008 را خواهید آموخت.

جلسه دو

کار های ساده با ویژوال استودیو

در این فصل می خواهیم شما را با برخی از کار های ساده با ویژوال استودیو آشنا کنیم.

همان طور که در قسمت قبل خواندید، برای شروع کار باید از نرم افزار ویژوال استودیو استفاده کنید. پس از نصب برنامه ، آن را باز می کنیم.



شکل 1-نمای اصلی برنامه

شکل 1 صفحه ی اولیه ی نرم افزار Microsoft Visual Studio 2008 می باشد. درپایین تو ضیحاتی بر طبق رنگ های اجزای عکس بالا داده شده است:

- لیست خطا ها(Eror List): تمامی خطا ها، پیام ها که در نوشتن برنامه به آن ها نیاز دارید ، در اینجا لیست می شود.دقت کنید:در بخش کد نویسی زمانی که جایی مشکل داشته باشد، زیر آن خط کشیده خواهد شد.

- نوار وضعیت : نواری است که در پایین پنجره قرار دارد و کارش نمایش وضعیت برنامه است که در عکس صفحه ی پیش کلمه ی Ready(آماده) نوشته شده است که به معنای این است که برنامه منتظر فرمان است.

- جعبه ابزار(Toolbox) : به پنجره ی سمت راست برنامه می گویند که در آن ابزار های مورد استفاده در هر پروژه موجود می باشد.

- راهنمای تازه کار ها(Getting Started) : به ستونی در صفحه ی شروع(Start Page) می گویند که در آن برای تازه کار ها به کتاب راهنمای نرم افزار لینک شده است.

- سازنده(Create): این یک گزینه برای ورود به پنجره ی ساخت پروژه است. در بخش ساخت پروژه ی جدید، توضیحات بیش تری داده شده است.

- باز کننده(Open) : این گزینه برای ورود به پنجره ی باز کننده ی پروژه هایی است که قبلا ساخته اید.

- پروژه های باز شده(Recent Project) : اگر قبلا پروژه ای را باز کرده باشید، آن پروژه در این جا قرار می گیرد که با کلیک بر روی نام آن، پروژه باز خواهد شد.

- اخبار: اخبار جدید از سازنده ی نرم افزار(مایکروسافت) در این جا قرار می گیرد.

- صفحه ی شروع(Start Page) : این اولین صفحه در لیست صفحات(بالای ستون اخبار) است. جزئیاتی که در بالا از قبیل اخبار،سازنده و... نام بردیم در این صفحه وجود دارد.

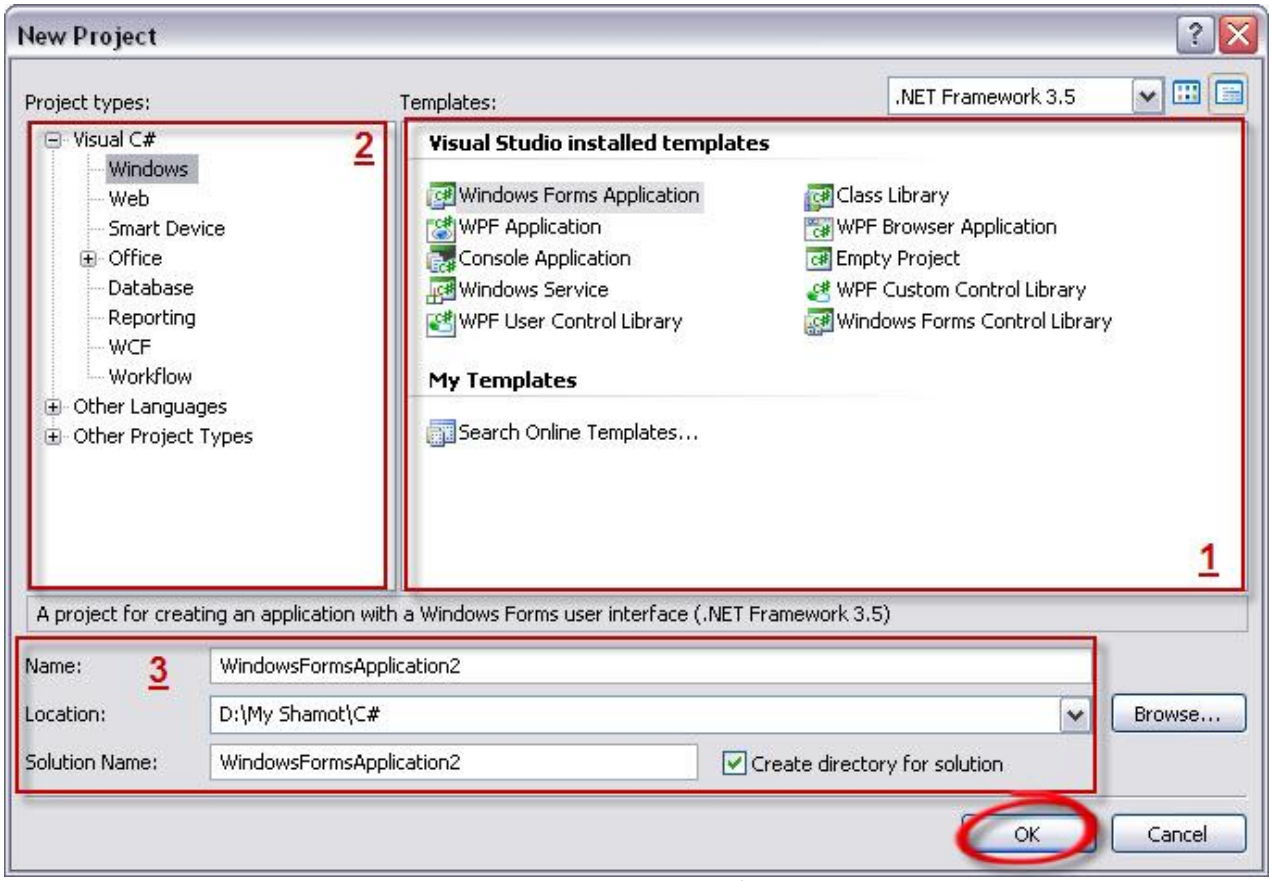
- نوار ابزار(Toolbar) : این نوار ابزار بر خلاف دیگر نرم افزار ها شامل فایل و ... نمی شود. گزینه های داخل آن هم به صورت جداگانه و در هر بخش مخصوصی گفته خواهد شد.

- نوار منو ها: این نوار شامل منو های File ، Edit ، View ، Tools ، Window ، Help که کاربرد همه ی آن ها واضح است و نیاز به تو ضیح ندارند.

با توجه به تو ضیحات بالا با اجزای اصلی برنامه آشنا شدید. حال اجزای حرفه ای تر نرم افزار را در بخش های لازم تفسیر می کنیم.

ابزار سازنده:

همان طور که مشاهده شد، در صفحه ی شروع ما یک عنصر به نام Create است که می توانیم با استفاده از لینکی که روبروی آن موجود است(Project)، به ابزار سازنده وارد شویم. در حقیقت شکل ظاهری این پنجره همانند شکل زیر می باشد:



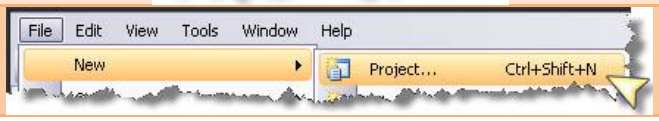
شکل 2-ابزار سازنده ی پروژه ی جدید

توضیحات شکل:

1. محل قرار گرفتن نوع و اشکال مختلف پروژه های جدید. مثل فرم(در واقع تمام نرم افزار هایی که ما از آن ها به طور روز مره استفاده می کنیم و کاملا برای کاربران عادی طراحی شده است می باشند.) و یا کنسول(که در بخش پیش با آن آشنا شدیم.)
2. همان طور که در قسمت پیش مشاهده کردیم، یک نمودار رشته ای وجود داشت. در واقع آن نمودار همین بخش دوم این پنجره است! در این جا می توانید تعیین کنید که نرم افزار شما در چه محیطی قرار است کار کند. در صورتی که زبان اصلی را هنگام نصب C# قرار نداده باشید، برای یافتن این زبان باید به قسمت Other Languages در همین قسمت مراجعه کنید.
3. مشخصات کلی برنامه را می توانید در این جا تعیین کنید. مشخصاتی مثل اسم پروژه، مکان ذخیره شدن، و فولدر پروژه را تعیین کنید.

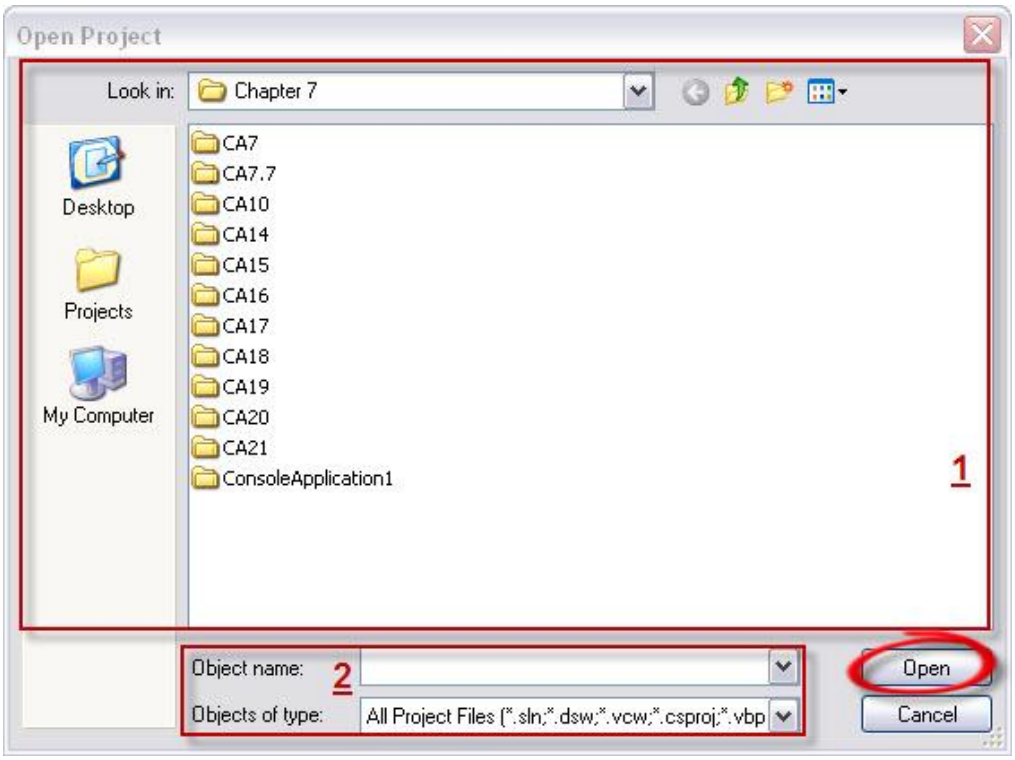
در آخر هم روی دکمه ی OK کلیک کنید تا پروژه ساخته شود.

برای باز شدن پنجره ی بالا چند راه اوجود دارد:

راه	شکل
1. از طریق صفحه ی شروع.	
2. از طریق نوار ابزار.	
3. از طریق منوی File	

ابزار باز کننده:

این ابزار برای زمانی است که شما می خواهید پروژه ای را که از قبل ساخته اید باز کنید. نمای کلی این پنجره به صورت زیر می باشد:

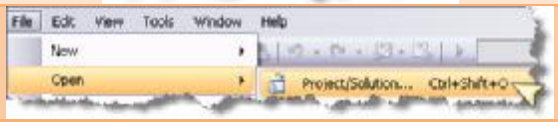


توضیحات پنجره:

1. قسمت مرورگر حافظه(Browser Of Memory): در این جا می توانید درون فولدر ها و اماکن را گشته و سپس پروژه را پیدا کنید و روی آن کلیک کنید.

2. این قسمت هم دو بخش دارد: یکی نام برنامه که اگر آن را می دانید باید در قسمت Object Name وارد کنید. برای دید بهتر و رده بندی توسط پسوند فایل ها هم از Object of type استفاده کنید.
در آخر هم بر روی Open کلیک کنید.

این پنجره هم مثل پنجره ی سازنده از راه های مختلفی باز می شود:

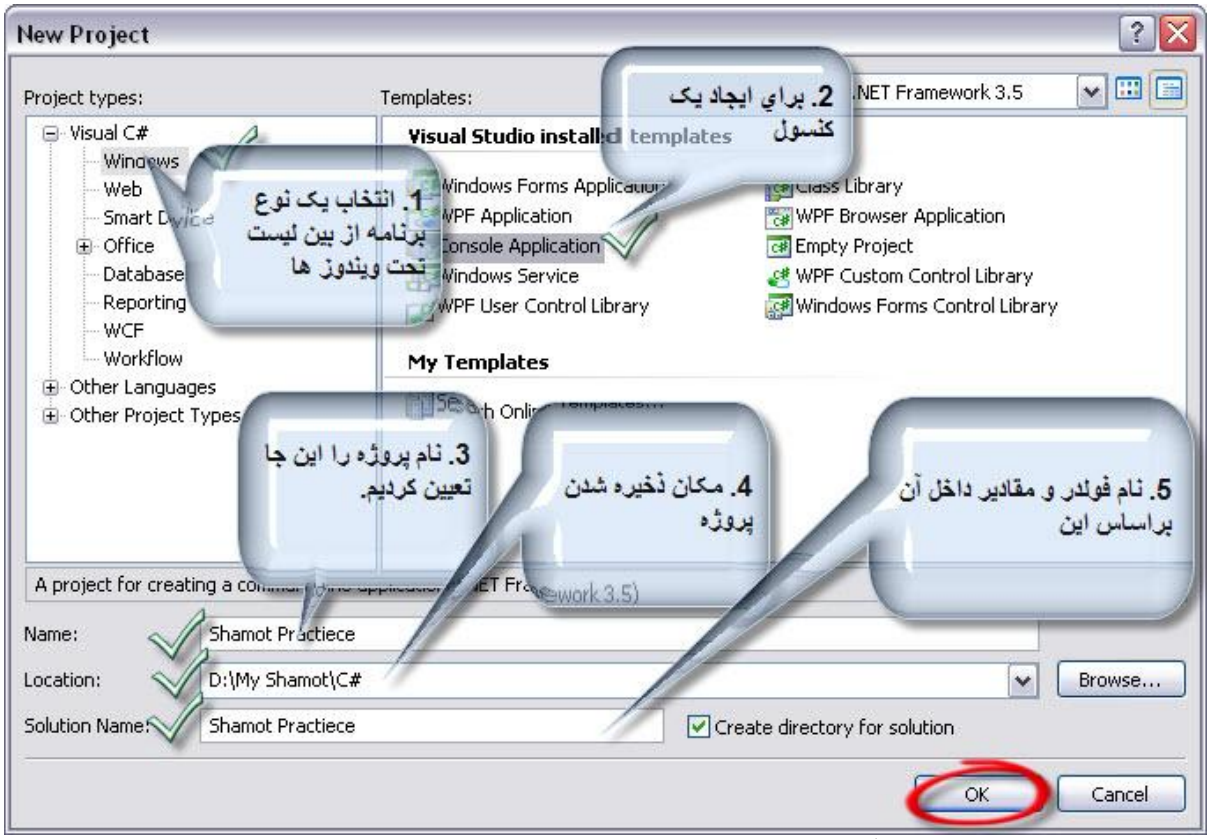
راه	شکل
1. از طریق صفحه ی شروع	
2. از طریق نوار ابزار	
3. از طریق منوی File	

خب حال نکات ساده ای را یاد گرفتیم! در قسمت بعدی اولین پروژه را می سازیم.

جلسه سه

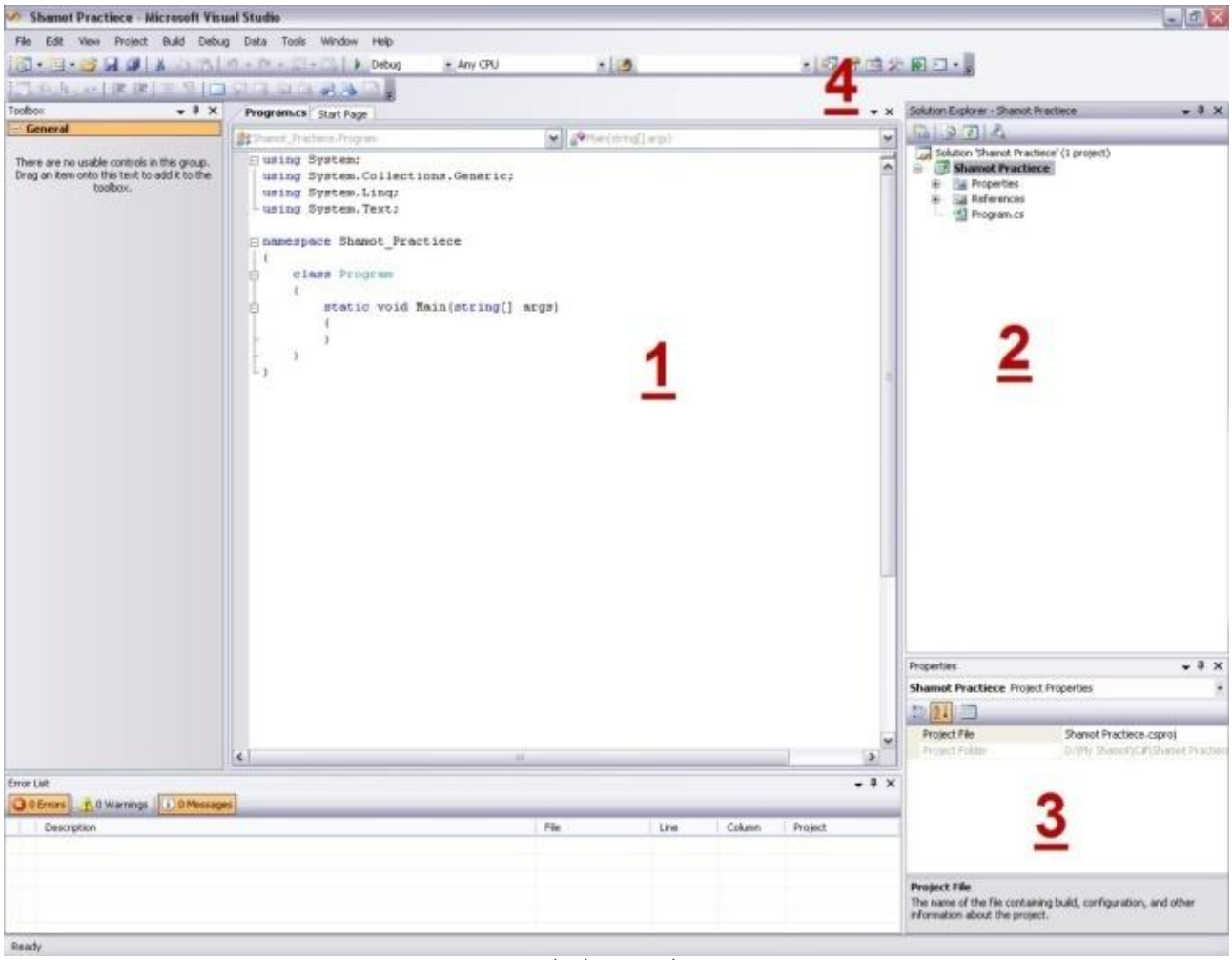
اولین کد نویسی

در قسمت پیش ، برخی کار های ساده را یاد گرفتید. حال باید وارد بخش کد نویسی شویم. برای این کار باید یک پروژه ی جدید ساخته شود. پس باید پنجره ی سازنده را باز کرد.من نام و مشخصات را در جاهای خود وارد می کنم.



شکل 1 – خلاصه کار های انجام شده برای ساخت یک پروژه ی جدید

خب پس از ساخت پروژه، نرم افزار این چنین می شود:



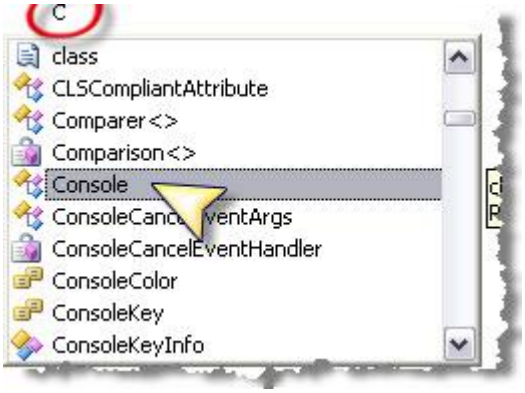
شکل 2 – شکل کلی پروژه

1. قسمت کد:در این قسمت کد های برنامه قرار دارد و شما می توانید در آن کد بنویسید.
2. مرورگر سلوشن(Solution Explorer): تمام اطلاعات در باره ی برنامه در این جا دیده می شود.
3. مشخصات(Properties): همه ی مشخصات برنامه(بسته به نوع برنامه) در این جا آرشیو شده اند و می شود در آن ها تغییر ایجاد کرد.
4. نوار لایه ها(LayerBar): تمام لایه ها و صفحاتی که به طور خودکار و یا از طریق Solution Explorer باز شده است، در این جا به صورت لیست شده قرار دارند.

یک امکان جالب که خیلی هم کاربردی است، منوی خلاصه است. این منو برای راحت و سریع تر شدن کارمان به وجود آمده است. زمانی که یک کد را دارید می نویسید برای مثال کد زیر:

```
Console.WriteLine("ShamotCity.vcp.ir+Shamot Group!+");
```

خب مطمئنا اگر بخواهیم نزدیک 100 خط کد به این درازی بنویسیم، کار مشکلی است. خب راه حلش هم همین منو است. با نوشتن یک حرف، منوی زیر می آید که می توان کد مورد نظر را از بین آن پیدا کرد:



شکل 3- منوی خلاصه

با کلیک کردن بر روی گزینه ی مورد نظر از درون منو، تمام کد به جای آن یک حرف می نشیند.

اصطلاحات و علامات مورد استفاده در C#:

- رشته(String): در واقع همان حروفی هستند که در کنار هم قرار می گیرند و متن را تشکیل می دهند. رشته ها مقدار عددی ندارند. حتی اگر درون آن ها عدد باشد و یا علامت های ریاضی. یعنی نمی شود بر روی آن ها عملیاتی انجام داد. در ضمن آن ها حتما باید درون " و " باشند.در غیر این صورت کد خوانده خواهد شد.
- عدد صحیح(Intiger): معنای آن که واضح است و در سی شارپ به صورت int خوانده می شود. اعداد درون هیچ علامتی نوشته نمی شوند و خواص اعداد صحیح را دارند.دقت کنید: اعداد صحیح و اعداد اعشاری و منفی ها درون آن ها قرار نمی گیرند.
- عدد گویا(Double):تنها فرقش با عدد صحیح در قرار گرفتن تمام اعداد است. یعنی همه ی اعداد به جز اعداد گنگ در آن قرار می گیرند.در اعشار هم باید به جای/ ، علامت . را گذاشت.
- علامت ; : این علامت در پایان تمام دستورات قرار می گیرد.
- نکته ی بسیار مهم:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Shamot_Practiece
{
    class Program
    {
        static void Main(string[] args)
        {
            //.کدتان را حتما باید این جا بنویسید
        }
    }
}
```

خب حال باید با اولین کد ها آشنا شوید. کد هایی ساده ولی بسیار کاربردی در کنسول.

یک سری کد ها در برنامه نویسی کنسول که به آن ها می توان آن ها ستون کنسول گفت خود کلمه ی کلیدی Console می باشد. این جور کد ها که اولشان این کلمه است، کد های اجرایی می باشند. یعنی کارشان تغییر در ظاهر برنامه است.

چاپ کردن(نمایش متن) متن:

```
Console.WriteLine("شما متن");
```

گرفتن(به معنای منتظر ماندن برای این که کاربر متنی را وارد کرده و اینتر را بزند)متن:

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

گرفتن کلید(منتظر ماندن برای زدن یک کلید):

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

چون کنسول پس از انجام کار خود، برنامه را می بندد. برای این که کاربر بتواند با صبر و حوصله کارکرد برنامه را نگاه کند، معمولا برنامه را از طریق کد ReadKey منتظر می گذاریم و کاربر با زدن یک دکمه، به برنامه فرمان می دهد که دکمه را زده است و چون برنامه دیگر منتظر کاری نیست، برنامه را می بندد.

خب البته نمی توان تمام کد های کنسول را معرفی کرد. ولی این کد ها مهم ترین آن ها بودند.

یکی از مهم ترین کار های نرم افزار ها حساب کردن و عملیات ریاضی است. سی شارپ می تواند 4 عمل اصلی را انجام دهد.

جمع=+ ، تفریق=- ، ضرب=* ، تقسیم=/

شما علامات را می توانید بین دو عدد قرار دهید.همان طور که در کد های ستون دیدید، کد WriteLine یک پرانتز دارد که درون آن دو"" است که شما می توانید حروف لاتین درون آن قرار دهید تا چاپ کند. حالا وقتی بخواهید جواب عملیاتی را چاپ کنید کافیست آن عبارت را درون پرانتز و بدون"" بنویسید.(چرا؟)

تمیز نویسی:

بهترین خصوصیت یک فرد برنامه نویس تمیز نویسی است. دقیقا مثل دست خط باید مراقب آن بود. اگر می خواهید از همین الان یک فرد متخصص در زمینه ی برنامه نویسی شوید، باید این اصل را رعایت کنید. همان طور که در کد های ویژوال استودیو می بینید، کد ها به ترتیب به صورت پلکانی از سر سطر دور شده اند و بعد نزدیک. این روش "دندانه نویسی" است که باعث نظم و ترتیب است. یک چیز دیگر: وقتی یک برنامه را می نویسید، باید از یک عنصر بسیار مفید به نام دیدگاه(Coment) استفاده کنید که کار آن نوشتن متنی است که در کد هیچ کاری نمی کند و خوانده نمی شود و جنبه ی توضیحاتی دارد. برای این که یک متن تبدیل به دیدگاه شود، باید // بگذارید وسپس متن را به دنبال آن بنویسید. فرض کنید که یک نرم افزار را ساخته اید و می خواهید نسخه ی جدید آن را بسازید. قطعا باید به کد های ورژن قبلی سری بزنید. حال اگر یادتان رفته باشد که هر کد چه کاری می کند، دیدگاه ها به یادتان می آورد. دیدگاه ها، از زبان شیرین فارسی پشتیبانی می کنند.

حال تمرینات زیر را بنویسید. ولی بدون نگاه کردن به جواب! ابتدا کد مورد نظر را بنویسید و سپس، آن ها را با جواب مقایسه کنید.

تمرین:

1. برنامه ای بنویسید که متنی را از کاربر بخواهد و پس از وارد کردن آن توسط کاربر، برایش چاپ کند:"Thank You!".
2. برنامه ای بنویسید که دو عدد 5687 را با 46857 جمع، تفریق، ضرب و تقسیم کند و سپس پاسخ ها را چاپ کند.

جلسه چهار

محل های ذخیره

در بخش پیش یاد گرفتیم که چگونه یک کد ساده بنویسیم و جواب چهار عمل اصلی را چاپ کنیم. و یا مقداری را بگیریم. هم چنین با تمیز نویسی و چند نکته در باره ی برنامه نویسی یاد گرفتیم. حال با گذرندان این فصل شما باید بتوانید:

- مفهوم محل های ذخیره را بدانید و آن ها را از هم تشخیص دهید.
- بتوانید نوع آن ها را تعیین کنید.
- به آن ها مقدار دهید.

محل های ذخیره، مکان هایی مجازی هستند که در آن ها اطلاعات ذخیره می شود. این محل ها به دو بخش کلی تقسیم می شوند:

1. *Ŭ*: مکان هایی که زمان اجرای برنامه تشکیل شده و زمان بستن آن ها خودشان و اطلاعات درونشان از بین می رود.
2. *Ŭäũ*: مکان هایی که اولین باری که برامه اجرا می شود تشکیل شده و سپس حتی پس از بستن برنامه اطلاعات حذف نمی شوند.

این محل های ذخیره جهت نگهداری و استفاده ی داده ها[1] و استفاده از آن ها در کد است. ما فعلا محل های ذخیره ی موقت را کار می کنیم.

انواع محل های ذخیره

- *Ŭ Y*[2]: ساده ترین و انعطاف پذیر ترین و همچنین پر کاربرد ترین محل ذخیره است. دقت کنید که محل های ذخیره دارای نام، جنس، نوع ومقدارند. پس در نتیجه آن ها باید تعریف شوند.حال متغیر ها را به صورت زیر تعریف می کنیم:

نام جنس = مقدار;

همانطور که خواندیم جنس ها انواع مختلفی هستند که مهمترین آن ها رشته، عدد صحیح، کارکتر، عدد گویا می باشند. نام هم باید حتما با حروف انگلیسی باشد و اول آن حرف باشد نه عدد. در ضمن نام باید عاری از هر گونه علامت باشد. فاصله هم در نام ها معنایی ندارد. یک عرف در برنامه نویسی وجود دارد که می گوید:«به خاطر این که نمی توان در نام گذاری از فاصله استفاده کرد، بهتر است اول هر کلمه در اسم را بزرگ بنویسیم.»باید دقت داشته باشید که متناسب با جنس یک متغیر، به آن مقدار دهید. مثلا اگر جنس

[1] در برنامه نویسی بیشتر به جای اطلاعات از این کلمه استفاده می شود.
[2] variable

متغیر رشته است، حروف را در "" بنویسید.لازم به ذکر است که مقدار دهی در همان زمان تعریف یک متغیرالزامی نیست. بنا براین می توان نوشت:

نام جنس ;

وقت آن رسیده که به طور عملی ببینیم:


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int Var1 = 0;
            int Var2;

            string Var3 = "Hello!";
            string Var4;

            Var2 = 1;
            Var4 = "Hi!";
        }
    }
}
```

طبیعی است که کلا در نامگذاری اشیاء نباید نام آن ها تکراری باشد.

§ ÿÿÚ[3]: زمانی ممکن است که شما احتیاج به مقدار زیادی متغیر داشته باشید(مثلا 100). آن موقع اگر بخواهید چندین متغیر بسازید، مطمئناً دچار مشکل می شوید و گیج شده و سپس در کد های پیچ در پیچ گیر می کنید. آرایه جدولی از متغیر ها است که در کنار هم و به صورت مجزا مقدار دهی می شوند.آرایه ها دارای جنس، بُعد، نام، تعداد ردیف در هر بعد هستند.

- جنس: انواع جنس را در بخش پیش خواندیم. جنس در اول عبارت تعریف یک آرایه می آید.
- بُعد: هر آرایه دارای بعد های مختلف است که هر چه ابعاد بیشتر باشد گنجایش ارایه بیشتر می شود.
- نام: نام هر آرایه می تواند یکی از مهم ترین عنصر های تعریف آن است. چون فقط با ذکر نام می توان مشخص کرد که می خواهیم از کدام آرایه ی تعریف شده در برنامه استفاده کنیم.
- تعداد ردیف در هر بعد: یک جور تعیین تعداد متغیر ها یا خانه های درون جدول آرایه است.

شمای تعریف و مقدار دهی یک آرایه ی 1 بعدی به صورت زیر است:

نام []جنس =new جنس[تعداد ردیف] ;

نام[اندیس[4]] = مقدار ;

شمای تعریف و مقدار دهی یک آرایه ی 2 بعدی به صورت زیر است:

[3] Array

[4] اندیس یا Index: به معنای شماره ی خانه ی مورد نظر

; [تعداد ردیف ,تعداد ردیف]جنس new= نام [,]جنس

; مقدار = [اندیس,اندیس]نام

ابعاد آرایه نامحدود است.جهت تعریف آرایه با ابعاد بالا تر کافیست روال بالا را ادامه دهید.اما یک مثال:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            //تعریف آرایه ها:
            int[] Array1 = new int[7];
            int[,] Array2 = new int[7, 7];
            string[, ,] Array3 = new string[7, 7, 7];
            string[,,,] Array4=new string[7,7,7,7];

            //مقدار دهی آرایه ها:
            Array1[0] = 55;
            Array2[0, 1] = 555;
            Array3[0, 1, 2] = "Hello!";
            Array4[0, 1, 4, 2] = "Hi!";

        }
    }
}
```

دقت کنید که در تعریف تعداد ردیف ها اعداد از 1 شروع می شوندولی در مقدار دهی و تعیین اندیس از 0.

توجه: ỷ ŬΎ ĶӃỷ ӊķ ӊ ỷӊự ỷӊķӊ ş Ķ Ÿ Ÿ Ŭ Ŭ Ķ Y Ÿ ũ ỷ ş Ă Ŭ Ÿ ӊķ ŏ Ÿ ũ Ÿ ķ Ÿ
Ÿ ũ ự ũ ӊự ỷ Ÿ . [5]. ĥ ķ.

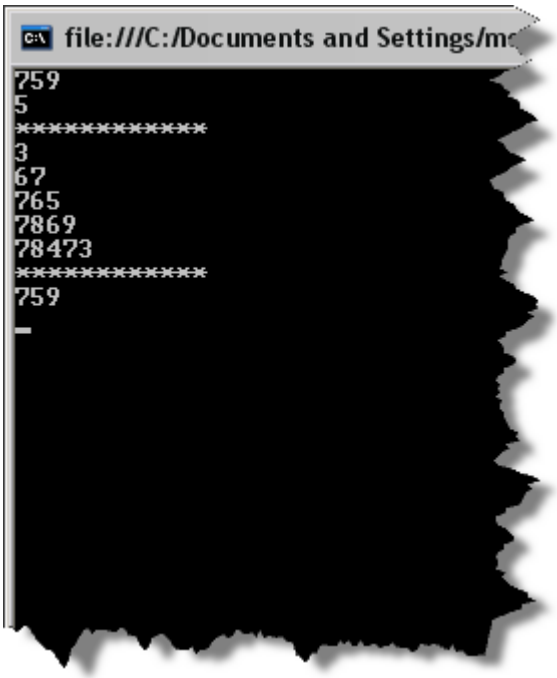
در مثال زیر درمی یابیم که چگونه می توان فرامین کنسول را در محل های ذخیره یا برعکس ریخت:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
```

[5] null

```
            //تعریف
            int Num1 = 5;
            int[] Num2=new int[5];
            string Num3;

            //دهی مقدار
            Num2[0] = 3;
            Num2[1] = 67;
            Num2[2] = 765;
            Num2[3] = 7869;
            Num2[4] = 78473;

            Num3 = Console.ReadLine();

            //چاپ
            Console.WriteLine(Num1);

            Console.WriteLine("************");

            Console.WriteLine(Num2[0]);
            Console.WriteLine(Num2[1]);
            Console.WriteLine(Num2[2]);
            Console.WriteLine(Num2[3]);
            Console.WriteLine(Num2[4]);

            Console.WriteLine("************");

            Console.WriteLine(Num3);

            Console.ReadKey();

        }
    }
}
```

نتیجه :



یک نکته: زمانی که خواستید جنسی را به جنسی دیگر مثلا رشته به عدد صحیح تبدیل کنید، می توانید از کد زیر استفاده کنید:

```
یک محل = Convert.ToInt32(String)
ذخیره از جنس عدد صحیح
```

یک نکته ی دیگر: جهت داشتن یک مقدار در یک رشته(مثلا دستور چاپ که رشته است) از عنصری به نام آرگمان استفاده می شود. این عنصر به شکل {} و یک عد درون آن است که در یک رشته می تواند مقداری را وارد کند که قابل تغییر است. بدین معنای که مثلا:

```
int Num1 = 2;
```

```
        int Num2 = 3;
        int Result = 5;


    Console.WriteLine("{0}+{1}={2}",Num1,Num2,Result);
```


جواب:

2+3=5

اگر در کد دقت کنید، به جای{0}، مقدار درون Num1 که اولین گزینه بعد ویرگول است.به جای {1} هم Num2 و به جای {2} همResult نشسته است.

تمرین:

1. برنامه ای بنویسید که دو عدد بگیرد و هر کدام را به صورت جداگانه در متغیر های Num1 و Num2 بریزد. سپس مجموع آن ها را در متغیری به نام Result بریزد. سپس Resultرا چاپ کند.

2. برنامه ای بنویسید که 5 عدد بگیرد و به ترتیب با 5 عدد 1، 2 ، 3 ، 4 ، 5 جمع کند. سپس جواب را چاپ کند.(با استفاده از آرایه)

جلسه پنج

دستورات شرطی

در جلسه ی پیش آموختیم که چگونه یک محل ذخیره بسازیم و چگونه از آن استفاده کنیم.حال با دستورات شرطی آشنا می شویم.

دستورات شرطی دستوراتی هستند که توسط آن ها می توانید شرط های منطقی بگذارید. بدین صورت که اگر آن شرط درست بود، کدی اجرا شود و اگر نبود آن کد اجرا نشود.این نوع شرط ها boolean نام دارند. اشیائی با جنس boolean همیشه یا مقدار true دارند و یا false.

True: به معنای این است که شرطی اجرا شده است. این مقدار فقط برای اشیاء منطقی(boolean) وجود دارد.

False: به معنای این است که شرطی اجرا نشده است. این مقدار فقط برای اشیاء منطقی(boolean) وجود دارد.

علامات شرطی:

- علامت ==: این علامت(که در واقع دو علامت را در دل خود دارد) به معنای این است که اگر یک مقدار برابر یک مقدار بود. مثلا مقایسه ی دو متغیر. دلیل بودن دوعلامت ==، در واقع به خاطر کاهش احتمال اشتباه کامپیوتر است. زیرا همانطور که در جلسه ی پیش خواندید، یک = به معنای این است که درون یک محل ذخیره را مقدار دهی کنیم. اما == یعنی اگر دو مقدار برابر بودند.
- علامت <: این علامت به معنای این است که اگر یک مقدار بزرگ تر از مقداری دیگر باشد. این مورد فقط برای اعداد مورد استفاده قرار می گیرد. (برای خواندن این عبارت می خوانیم: عبارت اول بزرگتر از عبارت دوم است.)
- علامت >: این علامت همان معنای علامت بالایی را میدهد. فقط مکان دو مقدار تغییر پیدا می کند.(برای خواندن این عبارت می خوانیم: عبارت اول کوچکتر از عبارت دوم است.)
- علامت =<: این علامت به معنای این است که اگر دو مقدار برابر یا اولی بزرگتر از دومی باشد(علامت را می خوانیم: بزرگتر مساوی)
- علامت =>: این علامت دقیقا برعکس علامت بالایی است.(علامت را می خوانیم: کوچکتر مساوی)
- علامت =!: به معنای این است که اگر دو مقدار برابر نباشند.(می خوانیم: مساوی نیست)

انواع دستورات شرطی:

- دستور if: دستور if به معنای اگر است. این دستور ساده ترین دستور شرطی است. یعنی اگر شرطی که ذکر شده است true(درست) باشد، کدی معلوم را اجرا کند و اگر false(نادرست) بود اجرا نکند.ساختار این دستور به صورت زیر است:

```
if (شرط)
{
    کدی که باید اجرا شود.
}
```

دستور if دارای کد های جانبی می باشد.مثلا زمانی که بخواهیم که بگوییم: اگر شرط false بود چه کار کن. در آن موقع از کد اضافی else استفاده می کنیم. دقت کنید که else باید بعد از دستور if باشد. و همینطور بیرون از {} دستور if. بنابراین:

```
if (شرط)
{
    کدی که باید اجرا شود.
}
else
{
    کدی که در صورت نادرست بودن شرط دستور بالا باید اجرا شود.
}
```

حال گاهی می خواهیم با ظرافت بیشتری برنامه بنویسیم. یعنی مثلا می خواهیم بگوییم: اگر شرط if برقرار نبود ولی این شرط برقرار بود، چه کار کن. در این صورت:

```
if (شرط)
{
    کدی که باید اجرا شود.
}
else if (شرط)
{
    کدی که در صورت برقرار نبودن شرط if اصلی، شرط else if را چک
    کند که برقرار است یا نه. اگر بود وارد آن شده و کد درون آن را اجرا
    کند.
}
```

- دستور swich: شاید استفاده از else if برای شرط های تعداد کم آسان باشد. اما اگر تعداد بالا برود، هم کدتان شلوغ می شود، هم قوانین تمیز نویسی را زیر پا گذاشته اید.راه حل این است که از عنصر جدیدی به نام Swich Case استفاده کنید. شکل کلی دستور:

```
switch (متغیری که قرار است شرط باشد)
{
    case مقدار مورد مقایسه:
        کدی که اگر متغیر ذکر شده در اول دارای مقدار بالا است
        باید اجرا شود.
        break;
```

```
            case مقدار مورد مقایسه:
کدی که اگر متغیر ذکر شده در اول دارای مقدار بالا است باید اجرا شود.

                break;
            default:
اگر هیچ یک از شرط های بالا برقرار نبود این کد اجرا شود.
                break;
        }
```


- دستور While: این دستور تا زمانی که شرط درست نشده باشد اجرا می شود. یعنی از این دستور خارج نمی شود تا بالاخره شرط true شود.(ممکن است هم که اصلا شرط true نشود. در این صورت می گوییم یک لوپ(حلقه) بینهایت است. البته این لوپ ها هیچگاه توصیه نمی شود. چون باعث بروز مشکل می شوند.)به طور کلی حلقه[6] ی While دو نوع است:

1. While: این نوع، همان While معمولی است. به همین دلیل هم یک اسم دارند.همان طور که می دانید، while باید کامپیوتر را وادار به صبر کند تا شرطش true شود. در صورتی که از while معمولی استفاده کنید.کامپیوتر ابتدا چک می کند که آیا شرط true است یا false. اگر true بود وارد حلقه شده و دستورات داخل آن را اجرا می کند.(داخل {}) اگر هم false بود، دوباره می آید سر شرط و چک می کند. تا وقتی که شرط false بود، این کار را می کند. شمای کلی:

```
while (شرط)
{
    کدی که باید درصورت برقرار بودن شرط اجرا شود.
}
```

یک مثال:

```
int i = 0;


while (i < 100)
{
    i++[7]
}
```

2. Do...While: فرق این نوع با آن شکل قبلی در این است که ابتدا کد داخل {} را اجرا می کند و سپس شرط را چک می کند.شمای کلی:

```
do
{
  کد
}
```

[6] حلقه: دستوری است که پیوسته اجرا می شود تا شرط برابر true شود.
[7] به معنای این است که به مقدار درون متغیر i یکی افزوده شود. به جای آن می توان نوشت:
i=i+1;

```
while (شرط);
```

یک مثال:


```
int i = 0;

do
{
    i++;
}
while (i < 100);
```

- دستور for: این دستور همانند دستور While، یک حلقه است. اما فرق آن با While این است که: در دستور for حتما یک متغیر باید مورد استفاده قرار گیرد.همچنین حتما باید آن متغیر در هر بار تکرار حلقه، تغییر کند.شمای کلی:

```
for (int i = 0; شرط; i++)
{
    کد
}
```

در صورتی که جایی لازم شد تا از یک حلقه خارج شوید، می توانید از تکه کد زیر استفاده کنید:

```
break;
```

این تکه کد را حتما باید بین {} متعلق به حلقه ی مورد نظر بگذارید.

تو در توها

گاهی ممکن است شما نیاز داشته باشید تا علاوع بر این که یک شرط را چک کند، شرط دیگری را بعد از وارد شدن به شرط اولی(دقت کنید: بعد از وارد شدن.)آن را چک کند. به این گونه کد ها تودرتو می گویند.ولی در کل استفاده ی زیادی از شرط های تودرتو، توصیه نمی شود. چون باعث در هم ریختگی کد شما می شود و این هم به نوعی نقض قوانین تمیز نویسی است.

در کل برای استفاده ی for های تودرتو، به دلیل این که نمی شود متغیر های همنام را ساخت، باید از متغیر های ناهمنام استفاده کرد. بنابراین معمولا متغیری که در for ساخته می شود(در())، به شکل زیر نامگذ اری می شود:

```
for (int i = 0; i < 100; i++)
{
    for (int j = 0; j < 100; j++)
    {
        for (int k = 0; k < 100; i++)
        {

        }
    }
```

```
}
```

بنابراین for اولی:i for دومی:j و for سومی:k. معمولا نباید بیش از 3 for تودرتو به کار ببریم.ولی در صورت نیاز عرفی برای آن در نظر گرفته نشده است.

هم اکنون تمرین های زیر را بنویسید و سپس آن ها را با جواب مقایسه کنید.

تمرین:

1. برنامه ای بنویسید که با استفاده از حلقه ی for، به اندیس های یک آرایه ی یک بعدی و 100 اندیسی، مقداری به ترتیب از 0 تا 99 را بدهد.

2. برنامه ای بنویسید که یک رشته از کاربر بگیرد و چک کند که اگر آن مساوی Shamot بود، چاپ کند:«Yes! Hello!». اگر مساوی Q بود، چاپ کند:«GoodBye!» و اگر هیچ کدام نبود، چاپ کند:« No! Repeat again!» این کار را 100 بار انجام دهد.

تمرین ها

پاسخنامه

جلسه دو:

پاسخ تمرین 1:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Shamot_Practiece
{
class Program
{
static void Main(string[] args)
{
//بنویسد سپس و بگیرد کاربر از رشته یک که است قرار برنامه این"Thank You!"
//رشته یک گرفتن برای:
Console.ReadLine();
//رشته نوشتن برای:
Console.WriteLine("Thank You!");

//نشود بسته برنامه که این برای:
Console.ReadKey();
}
}
}
```

پاسخ تمرین 2:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Shamot_Practiece
{
    class Program
    {
        static void Main(string[] args)
        {
            //تقسیم و ضرب تفریق، جمع، 46857 با را 5687 عدد دو باید برنامه
کند چاپ را ها پاسخ سپس و کند.

            //جمع:
            Console.WriteLine(5687 + 46857);
```

```
            //تفریق:
            Console.WriteLine(5687 - 46857);

            //ضرب:
            Console.WriteLine(5687 * 46857);

            //تقسیم:
            Console.WriteLine(5687 / 46857);


            //نشود بسته برنامه که این برای:
            Console.ReadKey();
        }
    }
}
```

جلسه 4:

تمرین 1:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int Num1;
            int Num2;
            int Result;

            //دهی مقدار
            Num1 = Convert.ToInt32(Console.ReadLine());
            Num2 = Convert.ToInt32(Console.ReadLine());

            Result = Num1 + Num2;

            //چاپ
            Console.WriteLine("{0}+{1}={2}",Num1,Num2,Result);

            //برنامه نگهداری
            Console.ReadKey();

        }
    }
}
```

تمرین 2:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] Nums=new int[5];
            int[] Results = new int[5];

            //دهی مقدار
            Nums[0] = Convert.ToUInt32(Console.ReadLine());
            Nums[1] = Convert.ToUInt32(Console.ReadLine());
            Nums[2] = Convert.ToUInt32(Console.ReadLine());
            Nums[3] = Convert.ToUInt32(Console.ReadLine());
            Nums[4] = Convert.ToUInt32(Console.ReadLine());

            Results[0] = 1 + Nums[0];
            Results[1] = 2 + Nums[1];
            Results[2] = 3 + Nums[2];
            Results[3] = 4 + Nums[3];
            Results[4] = 5 + Nums[4];

            //چاپ
            Console.WriteLine(Results[0]);
            Console.WriteLine(Results[1]);
            Console.WriteLine(Results[2]);
            Console.WriteLine(Results[3]);
            Console.WriteLine(Results[4]);

            //برنامه نگهداری
            Console.ReadKey();

        }
    }
}
```

جلسه 5:

تمرین 1:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Shamot_Practice2
{
```

```
    class Program
    {
        static void Main(string[] args)
        {
           int[] Nums=new int[100];

           for (int i = 0; i <100; i++)
           {
               Nums[i] = i;
               Console.WriteLine(i);
           }
           Console.ReadKey();

        }
    }
}
```

تمرین 2:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Shamot_Practice2
{
    class Program
    {
        static void Main(string[] args)
        {

           string YourText;


           for (int i = 0; i <= 100; i++)
           {
               YourText = Console.ReadLine();

               if (YourText=="Shamot")
               {
                   Console.WriteLine("Yes! Hello!");
               }
               else if (YourText=="Q")
               {
                   Console.WriteLine("GoodBye!");
                   break;
               }
               else
               {
```

```
                Console.WriteLine("No! Repeat again!");
            }
        }


    }
}
```

منابع

منابعی درباره ی C#

کلاس آموزش مقدماتی سی شارپ

سایت Techotopia http://www.techotopia.com