بسم الله الرحمن الرحیم

کتاب معلّم

(راهنمای تدریس)

برنامه‌سازی یک سی شارپ

رشتهٔ کامپیوتر

گروه تحصیلی کامپیوتر

زمینهٔ خدمات

شاخهٔ آموزش فنی وحرفه‌ای

۱۳۹۴

همکاران محترم و دانش‌آموزان عزیز :

پیشنهادات و نظرات خود را دربارهٔ محتوای این کتاب بـه نشانی تهران ـ صندوق پستی شمارهٔ ۴۸۷۴/۱۵ دفتر تألیف‌کتاب‌های درسی فنی و حرفه‌ای و کاردانش، ارسال فرمایند.

پیام‌نگار (ایمیل) **info@tvoccd.sch.ir**

وب‌گاه (وب‌سایت) **www.tvoccd.sch.ir**

وزارت آموزش و پرورش

سازمان پژوهش و برنامه‌ریزی آموزشی

برنامه‌ریزی محتوا و نظارت بر تألیف : **دفتر تألیف کتاب‌های درسی فنی و حرفه‌ای و کاردانش**

نام کتاب : **کتاب معلّم برنامه‌سازی یک‌سی‌شارپ**

مؤلفان : **مهناز کارکن، پردیس پیرایش، زهرا عسگری رکن‌آبادی، صدیقه رسولی، حسین صفوی، سارا اسماعیلی، سیدجعفر موسوی و محمدرضا یمقانی**

آماده‌سازی و نظارت بر چاپ و توزیع : **ادارهٔ کل نظارت بر نشر و توزیع مواد آموزشی**

تهران : خیابان ایرانشهر شمالی ـ ساختمان شمارهٔ ۴ آموزش و پرورش (شهید موسوی)

تلفـن : ۹ـ ۸۸۸۳۱۱۶۱، دورنگار : ۸۸۳۰۹۲۶۶، کد پستـی :۱۵۸۴۷۴۷۳۵۹،

وب سایت : www.chap.sch.ir

صفحه‌آرا : **فاطمه کارکن**

ناشر : **شرکت چاپ و نشر کتاب‌های درسی ایران : تهران ـ کیلومتر ۱۷ جادّهٔ مخصوص کرج ـ خیابان ۶۱ (دارو پخش)**

تلفن : ۵ ـ ۴۴۹۸۵۱۶۱، دورنگار : ۴۴۹۸۵۱۶۰، صندوق پستی : ۱۳۹ـ ۳۷۵۱۵

چاپخانه : **شرکت چاپ و نشر کتاب‌های درسی ایران «سهامی خاص»**

سال انتشار و نوبت چاپ : **چاپ اوّل، ۱۳۹۴**

شابک ۶-۲۴۸۸-۰۵-۹۶۴-۹۷۸ ISBN 978-964-05-2488-6

هدف کلی

ارائه راهبردهای تدریس و طراحی آموزشی کتاب برنامه‌سازی ۱

کتاب راهنمای معلّم برنامه‌سازی ۱ با دو رویکرد تألیف شده است :

۱ــ رویکرد دانشی

هدف : ارتقاء سطح دانش نظری و عملی هنرآموزان و ارائه مطالب موردنیاز هنرآموز، بالاتر از سطح کتاب درسی

۲ــ رویکرد روشی

هدف : افزایش مهارت تدریس و ارزشیابی از طریق شناخت اجزای طراحی آموزشی

فهرست

بخش اول : کلیات

بخش دوم : چارچوب فصل ها

۱. مفاهیم

۲. دانستنی هایی برای معلم

۳. واحد کار یادگیری

۴. ارزشیابی پیشرفت تحصیلی

۵. واحد کار عملی

بخش سوم : راهنمای فصل به فصل کتاب درسی

فصل ۱: آشنایی با مفاهیم پایه ای و پردازش داده ها

فصل ۲: آشنایی با زبان C #

فصل ۳: آشنایی با ویژوال استودیو

فصل ۴: آشنایی با انواع داده ها و متغیرها

فصل ۵: عبارت های محاسباتی

فصل ۶: دستورهای شرطی

فصل ۷: دستورات تکرار (حلقه‌ها)

سخنی با همکاران

کتاب حاضر تلاشی برای ارایه ی نکات مختلف آموزشی و تدریس درباره ی زبان برنامه نویسی C# است. در این کار کوشیده ایم چیستی، چرایی و چگونگی هر مطلب را به دقت بیان کنیم و علاوه بر عمق مفهومی به انگیزه و اندیشه همزمان توجه کنیم. بدون تردید برای یادگیری هر زبان برنامه نویسی، دانستن گرامر زبان به همراه ساختارهای آن مانند چگونگی ایجاد توضیحات، دستورات ساده و ترکیبی و نیز عملیات تعریف شده ضروری است، اما کاربردی کردن و استفاده از زبان در حل مسایل روزمره بسیار با اهمیت تر است.

در کتاب حاضر سعی بر این است که علاوه بر طرح نکات آموزشی به کاربرد نیز توجه شود و در نهایت هنرجو را به خلق برنامه های کاربردی علاقه مند و توانمند کنیم. ایجاد انگیزه برای استفاده ی مناسب از یک زبان برنامه نویسی، کاری دشوار ولی ضروری است و در این کتاب برآنیم تا از حافظه محوری در فرآیندیاددهی و یادگیری کار با زبان برنامه نویسی اجتناب کنیم.

از آنجایی که Help یک زبان برنامه نویسی، بهترین منبع یادگیری آن زبان محسوب می شود، آموزش کلمات کلیدی و واژگان به زبان انگلیسی ضروری و مفید است. ایجاد انگیزه برای استفاده از Help در بهره وری برنامه نویسی بسیار مهم و مفید است.

در آخر یادآوری می کنیم برنامه نویسی حلقه ی واسط بین مساله ها و نیازهای زندگی و نتایج دلخواه ما است و یادگیری پایه ای و مناسب آن در آینده ی این کار تاثیر گذار است. توجه به تعاریف، مفاهیم و کاربردها، می تواند زمینه ساز یادگیری موثر زبان برنامه نویسی شود.

در این کتاب تلاش شده است مواردی تدوین و گردآوری شود که موردنیاز یک هنرآموز است تا در فرایند آموزش این درس مورد استفاده قرار دهد. باتوجه به اینکه اولین کتاب راهنمای معلم رشته ی کامپیوتر در درس برنامه سازی یک تالیف شده است، قطعا خالی از اشکال نیست. ما را از نظرات سازنده خود بهره مند فرمایید. پیشاپیش از تمامی عزیزانی که با ارایه انتقادات و پیشنهادات خود ما را در بهبود این کتاب و تدوین راهنمای معلم دروس، راهنمایی می نمایند صمیمانه سپاسگزاریم.

RahnamaCSharp@hotmail.com

بخش اول :کلیات

۱. مقدمه

زندگی در تمامی مراحل خود دارای مساله های متعدد و متنوعی است. این مساله ها می توانند آسان یا مشکل باشند، اما نیاز به انتخاب راه حل مناسب برای آنها اجتناب ناپذیر است. با این رویکرد، زندگی سراسر تمرین حل مساله های گوناگون است. برای پیاده سازی حل مساله های مختلف، ابزارها و روش های متعددی، پیشنهاد شده است. یکی از فناوری های بسیار اثرگذار در بهبود زندگی، فناوری اطلاعات و ارتباطات و استفاده از رایانه است.
در برنامه ریزی نوین آموزشی کشور، آموزش یک زبان برنامه نویسی (C#) برای پیاده سازی در نظر گرفته شده است و تلاش شده است با استفاده از این زبان، توانمندی مورد نیاز برای هنرجویان تامین شود.

استفاده مناسب از زبان برنامه نویسی با آموزشی موثر، علمی وکاربردی امکان پذیرخواهدبود. درکتاب راهنمای فعلی ضمن بیان اهمیت و ضرورت استفاده از زبان C#، چگونگی یادگیری اجزای آن به صورتی کاربردی، تبیین می شود.

۲. معرفی درس برنامه سازی۱ (سی شارپ)

هر زبان برنامه نویسی، دارای اجزایی متناظر با نحوه ی حل مساله است. در حال حاضر پارادایم مسلط در حل مساله ها، **شی گرایی** است. هر مساله، در پایین ترین سطح خود شامل تعدادی **عملیات**، **شرط** و امور **تکراری** است. بنابر این می باید زبانی برای آموزش انتخاب شود که هم بتواند متناظر با پارادایم مسلط حل مساله، توانمند باشد و هم از نظر بازار کار قابل قبول و رایج باشد.

زبان C#، دارای هر دو ویژگی ذکر شده است. از طرفی با ایجاد مهارت تفکر شی گرایی، قابلیت های مورد نیاز را دارا است و از طرف دیگر ایجاد سیستم های نرم افزاری با این زبان، در بازار کار مقبول است. یادگیری این زبان، باعث می شود یادگیری سایر زبانها با این خصوصیات، با آسانی بیشتری امکان پذیر شود.

آموزش C#، به علت گستردگی آن در یک کتاب میسر نیست، بنابر این سه کتاب، برای آموزش مفاهیم و اجزای مربوط در نظر گرفته شده است. در کتاب برنامه سازی ۱، بیشتر به مفاهیم پایه و معرفی ساختار های اصلی و پایه ای پرداخته شده است. ایجاد برنامه های ساده برای یادگیری اولیه و آشنایی با مفاهیم پایه از اهداف مهم آموزش برنامه سازی ۱ است.

۳. اهمیت و جایگاه درس

درس برنامه سازی، در مجموعه ی آموزش رشته ی کامپیوتر از جایگاه بالایی برخوردار است. این درس به همراه دو کتاب برنامه سازی۲ و ۳ به هنرجو کمک می کند تا بتواند مساله هایی را در زندگی روزمره ی خود حل کند. یادگیری پایه ای و مناسب برنامه

سازی۱، به طور مستقیم و غیر مستقیم در یادگیری سایر درس ها نیز اثر گذار است و به کیفیت بخشی آموزش کمک می کند.

۴. انتظارات از برنامهٔ درس برنامه سازی۱

در درس برنامه ریزی۱، انتظار داریم هنرجو بتواند با استفاده از امکانات زبان برنامه سازی، حل یک مساله ساده را پیاده سازی کند و در عین حال در این درس، با امکانات مبنایی و پایه ای زبان **C#** ، آشنا شود. این موارد شامل یادگیری عملگر ها، ساختارهای تکرار و شرط است و با استفاده از آن ها، می توانیم پیاده سازی حل مساله های ابتدایی و ساده تر را انجام دهیم.

۵. چرا راهنمای معلم؟

کتاب راهنمای معلم برنامه سازی۱ با دو رویکرد دانشی – روشی نوشته شده است. در رویکرد دانشی، هدف ارتقای سطح دانش نظری و عملی هنرآموزان و ارایه مطالب مورد نیاز معلم، در حدود کتاب درسی و بالاتر از آن است. در رویکرد روشی، هدف افزایش مهارت تدریس و ارزشیابی از طریق شناخت اهداف درس و روش های پیشنهادی است. بدون تردید کتاب راهنمای معلم، اهداف، دلایل و چگونگی تدریس را - به صورت یک پیشنهاد - بیان می کند و امیدواریم با دریافت بازخوردهای مرتبط به تدریس و آموزش موثر این درس کمک کنیم.

بخش دوم : چارچوب فصول

در تمامی ۷ فصل کتاب راهنمای معلم برنامه سازی یک، چارچوب زیر تا حدامکان رعایت شده است.

۱. مفاهیم

در هر فصل در بخش مفاهیم، به تعاریف اولیه و هدف کلیِ فصل پرداخته شده است.

- **کلیدواژه های فصل**

این کلیدواژگان کمک می کند تا همکاران گرامی بدانند در این فصل با چه مطالبی روبرو هستند. به ویژه برای استفاده در موتورهای جستجوی تحت وب مناسب هستند.

- **تعاریف**

هریک از کلیدواژگان تعریف شده اند تا با توجه به تعریف برخی واژه ها در متن، به درستی مشخص شود منظور از این واژه چیست و چه تعریفی مورد نظر مولف است.

- **کاربرد**

دانستن واژگان هر موضوع، دامنه و حدود آن موضوع را روشن و آشکار می کند. این آشنایی به درک عمق و وسعت مفاهیم و در نتیجه استفاده ی عالمانه در آن حوزه

منجر می شود.

- **چرایی**

بی تردید نمی توان از درک یک موضوع بدون دانستن کلمات کلیدی آن حوزه، صحبت کرد. واژگان هر حوزه به تعمیق شناخت، یادگیری و بار فنی و ذهنی موضوع کمک می کند.

۲. دانستنی هایی برای معلم

در بخش دانستنی ها تاکید بر این است که مخاطب معلم است نه دانش آموز! بنابراین باید توجه کرد که هیچ یک از مفاهیم این بخش مناسب کلاس درس نیستند و معلم نباید آن ها را ملاک عمل قرار دهد. در این بخش تلاش شده است با ارایه مطالب عمیق تر، به آمادگی معلم برای حضور با قدرت علمی بالاتر کمک کند تا بتواند فراتر از سطح کتاب درسی دانش و مهارت خود را افزایش دهد.

- **تاریخچه**

دانستن تاریخچه ی به وجود آمدن یک روش، مهارت یا ابزار ارتباط معنا داری با نیاز بشر برای ایجاد آن دارد. به عبارت دیگر وقتی سخن از زمان بندی یا تاریخچه می شود نا خودآگاه ضرورت ایجاد یا کشف یک ابزار مورد نظر است.

- **عمق دهی به موضوع**

مناسب است در آموزش، معلم نسبت به مفاهیم یا ابزارهای مورد تدریس آگاهی و دانش عمیق تری داشته باشد. هر چه معلم در موضوع تدریس خود بیشتر، مناسب تر و عمیق تر بداند هم در مواجهه با سوال و هم در تدریس خود مفید تر و موثر تر ظاهر می شود.

- **مزایا و معایب**

دانستن مزایا و معایب هر یک از ساختارها و اجزای زبان برنامه سازی به استفاده ی صحیح آن ها کمک می کند و در نتیجه کد کارآمد تری ایجاد می شود.

- **مفاهیم یا جدول های تطبیقی با زبان های دیگر**

بدون تردید مقایسه ی زبان های برنامه سازی و اجزای آن ها به انتخاب صحیح زبان برنامه سازی کمک می کند. این کار با ایجاد جدول های مقایسه ای بسیار به شناخت معلم کمک می کند.

- **نکات مهم**

بیان نکات کلیدی و با اهمیت در هر حوزه ی آموزش برنامه سازی به تعمیق شناخت معلم از مبحث کمک می کند.

در این بخش، پرسش هایی جمع آوری شده که ممکن است در بخش دانستنی هایی برای

معلـم بـه آن پرداختـه نشـده باشـد کـه یـا در ذهـن پرسشـگر معلـم ممکـن اسـت ایجـاد شـده باشـد یـا دانـش آمـوزی بـا تکیـه بـر مطالعـات قبلـی از معلـم سـوال کنـد و چالـش ایجـاد نمایـد.

۳. واحد کار یادگیری

دریک واحد کار یادگیری، ما به ۷ پرسش یک معلم در فرایند تدریس پاسخ می دهیم:

۱.	چه مطلبی را می خواهم تدریس کنم؟	موضوع
۲.	به چه منظوری می خواهم تدریس کنم؟	اهداف آموزشی
۳.	در چه زمانی می خواهم تدریس کنم؟	طرح درس روزانه- سالانه
۴.	به چه کسانی می خواهم تدریس کنم؟	رفتار ورودی- پیش نیاز
۵.	با چه روشی می خواهم تدریس کنم؟	روش تدریس
۶.	با چه وسیله ای می خواهم تدریس کنم؟	تکنولوژی آموزشی
۷.	آیا در کار خود موفق هستم؟	ارزشیابی

تعریف واحد کار یادگیری

واحـد کار از نـگاه مولفـان سـناریویی بـرای آمـوزش اسـت کـه مـی توانـد اهـداف طـرح درس را نیـز پوشـش دهـد. در واحـدکار سـعی شـده اسـت بخـش هایـی از طـرح درس کـه اولویـت بالاتـری دارنـد حفـظ شـود و گزیـده ای از مولفـه هـای یـک طـرح درس بـا انسـجام مناسـب کنـار هـم گـرد آینـد تـا:

۱- با حفظ بخش ها و مولفه های مهم یک طرح درس، جامعیت یک واحد کار[۱] خدشه دار نشود.

۲- در زمانِ نوشتنِ واحدِ کار صرفه جویی شود.

۳- الگوی ساده ای مشابه طرح درس ارایه شود.

۱ واحدکار=یک یا دو جلسه آموزشی

تفاوت واحد کار با طرح درس

نخستین بار طرح درس در سال ۱۹۱۸ توسط " فرانکلین بوبیت " مطرح شد و کاربرد آن در فرایند یاددهی – یادگیری مورد تأکید واقع شد. طرح درسی که امروزه مورد استفاده قرار می گیرد، در سال ۱۹۵۰ به وسیله روان شناسان برجسته امر تعلیم و تربیت بنجامین. اس.بلوم[۲] شکل گرفت.

طرح درس، برنامه ریزی و سازمان بخشیدن به فعالیت هایی است که برای زمان مشخصی معلم با توجه به اهداف و محتوای درس و مخاطب تدوین می کند. یک طرح درس دقیق و علمی دارای سه بخش زیر است:

۱- فعالیت های قبل از تدریس (مرحله طراحی)

۲- فعالیت های حین تدریس (مرحله اجرا)

۳- فعالیت های بعد از تدریس (مرحله ارزشیابی)

در واحد کار از هر سه فعالیت، بخش هایی گزیده و تکمیل شده اند.

- **نمونه ها**

در بخش نمونه ها تلاش شده است از مطالب اصلی کتاب درسی که چالش بیشتری در فرایند تدریس ایجاد می کند، واحد کار نمونه آورده شود. با توجه به تنوع موضوعی انتخاب شده در نمونه ها، تنوعی از ارزشیابی، تمرین و روش های تدریس با ذکر نمونه ها بیان شده است.

- بودجه بندی ساعت تئوری-عملی

بودجه بندی یا طرح درس سالانه یا کلی عبارت از این است که محتوای یک ماده درسی بر اساس هدف و برای یک سال آموزشی، به مراحل و قدم‌های مناسب و مشخص تقسیم شود.

- **اهداف فصل**

در واحد کار سعی شده است چند نوع هدف از اهداف فصل مورد توجه قرار بگیرد که عبارتند از: اهداف رفتاری، هدف کلی، اهداف پنهان درس، هدف ایده آل.

---

۲ بنجامین بلوم(۱۹۱۳-۱۹۹۹). دکترای تعلیم و تربیت-عضو ستاد آزمون های دانشگاه شیکاگو. او به عنوان معلم، محقق، استاد و مشاور تربیتی مسوولان تعلیم و تربیت کشورهای مختلف بود. از وی هیجده عنوان کتاب به چاپ رسیده که تعدادی از آن ها به زبان فارسی نیز ترجمه شده است. یکی از دلایل معروفیت بلوم « طبقه بندی هدف های تربیتی» است. بلوم با همکاری تعدادی از همکاران خود، هدف های آموزشی را (هدف هایی که دانش آموزان پس از طی کردن یک دوره آموزشی می توانند به آن ها دست یابند) طبقه بندی کرد. طبقه بندی آن ها بسیار مشهور است. در این طبقه بندی، هدف های آموزشی در سه حیطه ی حوزه شناختی، عاطفی و روانی - حرکتی تقسیم بندی شده است.

۴. ارزشیابی پیشرفت تحصیلی

پیش از تعریف ارزشیابی پیشرفت تحصیلی، بهتر است با ادبیات ارزشیابی آشنا شویم. واژه های اندازه گیری، آزمون، سنجش و ارزشیابی، هریک مفاهیم جداگانه ای دارند.

الف.اندازه گیری

اندازه گیری یک فرایند است که در آن ویژگی ها یا صفات اشیاء و افراد تعیین شده و به صورت عدد و رقم گزارش می شود. این فرایند نیاز به وسیله ی اندازه گیری دارد. وسیله ی اندازه گیری مستقیم ویژگیهای فیزیکی، مثل طول یا قد و وزن، متر و ترازو است.

ب.آزمون

علاوه بر ویژگیهای فیزیکی، رفتارهای آشکار افراد مثل تعداد غیبت های دانش آموز را نیز می توان به طور مستقیم اندازه گیری کرد اما برخی ویژگی ها مانند هوش، خلاقیت، یادگیری و مانند این ها را نمی توان به طور مستقیم اندازه گیری نمود و متداول ترین شیوه اندازه گیری غیرمستقیم این ویژگی ها، آزمون است. آزمون وسیله یا روشی نظامدار برای اندازه گیری نمونه ای از رفتار است. پس آزمون وسیله ی اندازه گیری یادگیری است.

ج.سنجش

درسنجش وسایل، فنون و روشهای مختلفی مانند آزمون، پرسشنامه، پروژه، مصاحبه و... برای جمع آوری اطلاعات به کار می رود که از آزمون مفهوم گسترده تری دارد.

ارزشیابی پیشرفت تحصیلی

ارزشیابی به یک فرآیند نظام دار برای جمع آوری، تحلیل و تفسیر اطلاعات گفته می شود به این منظور که تعیین شود آیا هدف های مورد نظر تحقق یافته اند یا در حال تحقق یافتن هستند و به چه میزانی؟ یکی از ویژگیهای مهم ارزشیابی تعیین کیفیت است. در فرایند ارزشیابی داوری ارزشی با توجه به کیفیت صورت می گیرد. در ارزشیابی آموزشی یک برنامه درسی، ابتدا هدف های کلی برنامه درسی تعیین می شود. در ارزشیابی پیشرفت تحصیلی، کیفیت را به عنوان میزان دستیابی دانش آموزان به دانش مهارتها و توانائی هائی که از آنها انتظار می رود تعریف کرده اند.

ارزشیابی= اندازه گیری + قضاوت

در ارزشیابی می خواهیم بدانیم:

۱- آیا اهداف موردنظر تحقق یافته اند؟

۲- کیفیت چه میزان بوده است؟

برای پاسخ به این دو سوال، باید دو اقدام اساسی انجام شود:

۱) اهداف آموزشی از پیش تعیین شوند.

۲) عملکرد یادگیرندگان را با ابزار مناسب اندازه گیری شود.

در ادامه به هر دو اقدام به صورت مفصل خواهیم پرداخت.

اهداف آموزشی(اهداف یادگیری)

در آموزش و پرورش دو نوع هدف وجود دارد: اهداف کلی و اهداف دقیق. اهداف کلی به وسیله برنامه ریزان در سطح وزارتخانه تهیه می شود و اهداف دقیق و عینی اهدافی هستند که معلمان با توجه به اهداف تهیه می کنند و از روی این اهداف سنجش صورت می گیرد. به هدف های دقیق آموزشی، اهداف رفتاری نیز می گویند که نشان دهنده ی تغییرات حاصل در رفتار یا عملکرد فراگیران است. امتیاز مهم اهداف رفتاری این است که به سهولت قابل شناسایی و اندازه گیری هستند و معلم منظور و مقصود خود را از آموزش به طور دقیق مشخص می کند و به فراگیران می گوید که چه انتظاراتی از آنان دارد.

نوشتن هدف های رفتاری کامل، کار تهیه سوال های امتحانی را به غایت آسان می کند، زیرا آنها به معلم می گویند که در نوشتن هر سوال چه نوع عملکردی را از دانش آموزان در نظر داشته باشد، چه شرایطی برای پاسخ دهی دانش آموزان فراهم آورد و با چه معیارها یا ملاک هایی کار آنان را ارزشیابی کند.(سیف، ۱۳۸۴، ص۱۲۸)

طبقه بندی هدف ها ی آموزشی

از معروف ترین طبقه بندیِ هدفهای آموزشی طبقه بندی بنجامین. اس. بلوم است. در این طبقه بندی هدفهای آموزشی به سه طبقه کلی به نام های حوزه شناختی، عاطفی و روانی-حرکتی تقسیم شده اند.

حوزه شناختی اهداف آموزشی

هدفهای حوزه شناختی به جریان هایی که با فعالیت های ذهنی و فکری آدمی سروکاردارند مربوط می شوند. این اهداف را به دو طبقه ی اصلی دانش و فرادانش تقسیم می کنیم. طبقه دانش دارای سطح **دانش** و طبقه فرادانش دارای پنج سطح دیگر است که شامل **فهمیدن، به کاربستن، تجزیه و تحلیل، ترکیب و ارزشیابی** است.

- **حوزه شناختی**

سطوح حیطه شناختی از ساده تا پیچیده عبارتند از:

۱- دانش

یادآوری – بازخوانی و بازشناسی امور

مثال: تعریف کند... ذکر کند... نام ببرد...

۲- فهمیدن

توانایی درک منظور یا مقصود یک مطلب.

مثال: **دلایل را برشمارد... مثالهایی ذکر کند... ترجمه کند(عربی و انگلیسی)...**

۳- به کار بستن

توانایی استفاده از امورانتزاعی، قواعد واصول و روشها در موقعیتهای عینی و عملی.

مثال: مسائل ریاضی را حل کند... بکاربندد... نمونه ای رابرای خودش تعیین کند...

۴- تحلیل

توانایی شکستن یک مطلب یا موضوع به اجزا یا عناصر تشکیل دهنده آن.

مثال: اجزا را مشخص کند.. هدف اصلی یک مطلب را تشخیص داده و بیان کند... عقاید را از واقعیتها متمایز سازد...

۵- ترکیب

توانایی پهلوی هم گذاشتن عناصر و اجزا برای ایجاد یک الگو یا ساختار نو.(آفرینندگی و خلاقیت)

مثال: برنامه کامپیوتری بنویسد... انشا بنویسد...

۶- ارزشیابی

توانایی قضاوت یا داوری کمی و کیفی درباره امور، با توجه به ملاکهای معین.

مثال: نقدوبررسی کند... داوری کند...

حوزه عاطفی اهداف آموزشی

حوزه عاطفی شامل هدف هایی است که به علایق، احساسات، ارزش ها، اخلاقیات و عواطف مربوط می شود و سطوح آن عبارتند از: **توجه کردن، واکنش نشان دادن، ارزش گذاری، سازمان بندی کردن، متبلور ساختن.**

۱- توجه کردن(دریافت)

نسبت به اهمیت یادگیری تمایل نشان دهد و با دقت به فعالیت های کلاس توجه کند

مثال: می پرسد ...انتخاب می کند ...دنبال می کند...گوش می کند ... سؤال می- کند ...

۲- واکنش

از مطالعه لذت می برد، برای انجام کارهای ویژه داوطلب می شود

مثال: موافقت می کند ،کمک می کند ،پاسخ می دهد

۳- ارزش گذاری

به نقش معلم در زندگی روزمره ارج می نهد، نسبت به پیشرفت جامعه احساس تعهد می کند.

مثال: پیش قدم می شود...تحسین می کند...تشکر می کند...دعوت می کند...

۴- سازمان بندی ارزش ها

نقش برنامه ریزی منظم در حل مسائل را می شناسد، مطابق علایق، توانایی ها و اعتقادات خویش برنامه ای برای زندگی خود تنظیم می کند.

مثال: طرفداری می کند، اصلاح می کند... منظم می کند...تعمیم می دهد... تعدیل می کند...

۵- تبلور ارزش ها

برای انجام کار مستقل از خود اعتماد به نفس نشان می دهد، عادات خوب بهداشتی را حفظ می کند.

مثال: تحت نفوذ قرار می دهد... ممیزی می کند... خدمت می کند... تجدید نظر می کند...

حوزه روانی- حرکتی اهداف آموزشی

حوزه روانی – حرکتی به زمینه ی مهارت های حرکتی یا حرکات بدنی مربوط است. بنابراین هر فعالیتی که علاوه بر جنبه روانی دارای جنبه جسمانی هم باشد، در این حوزه جای دارد.(سیف، ۱۳۸۴) فعالیت های این حوزه علاوه بر اینکه در حوزه شناختی قرار دارند و ذهن را درگیر می کنند، به حوزه عاطفی نیز مربوط می شوند. در این حیطه فعالیت ها منجر به افزایش توانایی عضلات و ایجاد هماهنگی بین آنها می شود. به زبان دیگر این بخش از هدف های آموزشی به گونه ای است که انجام آنها نیازمند همکاری اعصاب (سیستم عصبی) و ماهیچه هاست. مهارت هایی مانند: نوشتن، دویدن، خیاطی کردن، رانندگی، کار آزمایشگاهی، نقاشی و... در این حوزه می باشند. این حیطه دارای طبقه بندی های مختلفی است که به دلیل اهمیت آن ها در دروس کارگاهی و عملی شاخه فنی و حرفه ای و کاردانش، همه آنها آورده می شود.

- **طبقه بندی دوه از حوزه روانی - حرکتی**

دوه هدف های تربیتی حیطه روانی – حرکتی را در پنج سطح تقسیم بندی کرده است:

۱. تقلید: تقلید، پایین ترین سطح مهارتهای حرکت است. در این سطح فرد بدون کمک و راهنمایی دیگران نمی تواند کاری را که جنبه حرکتی دارد، درست انجام دهد.

مثال: نوشتن به کمک مربی، انجام حرکات رانندگی با راهنمایی مربی[۳]

۲. اجرای مستقل: در این سطح، یادگیری مهارت اندکی از مرحله قبل بالاتر است، به طوری که فرد میتواند بدون کمک دیگران عملی را انجام دهد.

۳. دقت: توانایی انجام مهارت حرکتی هنگامی به این سطح می رسد که فرد بتواند کاری را به درستی و با دقت، سرعت و ظرافت کافی انجام دهد و تکرار نماید. مثال: کاربرد مواد شیمیایی با رعایت نکات ایمنی، سرخوردن روی یخ ، بافتن گلهای قالی.

۳ مربیان تعلیم رانندگی یک پدال ترمز جلوی پای خود دارند تا در موقع لزوم از آن استفاده کنند.

۴. هماهنگی حرکات: هنگامی که فرد به این سطح از مهارت میرسد، می‌تواند چند فعالیت را به طور همزمان و هماهنگ انجام دهد. بعنوان مثال دست ها، پاها ، چشمان خیاط ماهری که برای دوختن لباس ازچرخ خیاطی پایی استفاده می کند... تسلط در شنا کردن... مهارت در رانندگی با اتومبیل...

۵. عادی شدن: این سطح بالاترین مرحله یادگیری و کسب مهارت در حیطه روانی – حرکتی است. طی آن انسان به طور خودکار به انجام کارهای دقیق و موزون عادت میکند. دراینجا، دیگر شخص برای انجام یک عمل به فکر کردن و صرف انرژی عصبی جهت هماهنگ کردن فعالیتها و توالی آنها نیاز ندارد و اغلب فعالیتهای او بدون شک و تردید، به طور ناخودآگاه صورت میگیرد. مثال: رانندگی می کند... شنا می کند...

از نظر دوه سه نوع آمادگی برای انجام مهارت های حرکتی ضروری است:
الف – آمادگی ذهنی ب- آمادگی جسمی ج- آمادگی عاطفی

خلاصه سه حیطه

بنابراین معلمان پس از آموزش یک مطلب یا انجام فعالیت های تربیتی انتظار دارند که دانش آموزان در پایان در رابطه با حیطه شناختی مطالبی را بدانند، بفهمند، بکاربرند، تجزیه وتحلیل کنند، ترکیب نمایند و یا ارزشیابی کنند. در رابطه با حیطه عاطفی نسبت به مسئله، پدیده ها و رویدادها، توجه داشته باشند، واکنش نشان دهند، ارزش گذاری کنند، سازمان بندی کنند، و ارزش ها در شخصیت آنها تبلور یابد و در رابطه با حیطه روانی- حرکتی مهارت هایی در سطوح تقلید، اجرای مستقل، دقت، هماهنگی حرکات و عادی شدن کسب کنند.(سیف، ۱۳۸۴)

• جدول مشخصات آزمون

همان طوری که قبلا اشاره شد، دومین اقدام برای ارزشیابی پیشرفت تحصیلی، سنجش عملکرد فراگیران توسط ابزار آزمون است. مهمترین گام در تهیه ی آزمون های پیشرفت تحصیلی، تهیه جدول مشخصات برای موضوعی است که قرار است آزمون آن تهیه شود. جدول مشخصات یک جدول دوبعدی شامل هدف و محتواست که در صورت دقیق و کامل بودن می تواند به روایی و اعتبار آزمون کمک کند.

نوشتن هدف های رفتاری کامل کار تهیه سوالات امتحانی را آسان می کنند زیرا آنها به معلم می گویند که در نوشتن هر سوال چه نوع عملکردی را از دانش آموزان در نظر داشته باشد چه شرایطی برای پاسخ دهی دانش آموزان فراهم آورد و با چه ملاکهایی کار آنان را ارزشیابی کنند.

چون تعداد اهدافی که در جدول قرار داده می شود محدود بوده و تنها نمونه ای از اهداف

آموزشی کتاب در آن درج می شود، هر هدف باید به گونه ای انتخاب شود که چند هدف کوچکتر را پوشش دهد.(حشمتی، ۱۳۸۴، ص۸۳) یک آزمون خوب پیشرفت تحصیلی آزمونی است که به بهترین شکل منعکس کننده ی تمامی هدف های آموزشی باشد. (سیف، ۱۳۸۴) اما تهیه چنین آزمونی غیرعملی است، بنابراین آزمونی خوب است که حاوی نمونه ی درستی از هدف ها و محتوای درس باشد تا از روایی بالایی برخوردار باشد. جدول مشخصات آزمون این نقش را ایفا می کند تا از میان این همه اهداف درس، نمونه ی مناسبی انتخاب شود. بنابراین هدف از تهیه جدول مشخصات آزمون این است که:

- کمک می کند از مجموع بزرگی از سوالات ممکن، نمونه ای خوب انتخاب شود.
- نمونه ی انتخابی، به خوبی معرف محتوا و اهداف درس باشد.
- به ایجاد تعادل بین آموزش و سنجش کمک شود.

بعد محتوا	بخش۱			بخش۲				بخش۳
بعد هدف	فصل ۱	فصل ۲	فصل ۳	فصل ۱	فصل ۲	فصل ۳	فصل ۴	
هدف های حوزه شناختی	تعداد و نوع سوال			تعداد و نوع سوال				
هدف های حوزه عاطفی	تعداد و نوع سوال			تعداد و نوع سوال				
هدف های حوزه روانی-حرکتی	تعداد و نوع سوال			تعداد و نوع سوال				

نمونه آزمون مبتنی بر جدول مشخصات

دراین کتاب، دو جدول مشخصات به عنوان آزمون های دوره ای چند فصل تهیه شده و متناسب با این جدول، آزمونی طراحی و ارایه شده است. این جدول می تواند برای طراحی آزمون های مشابه بارها و بارها به کار برده شود.

- **ارتباط تمارین فصل و اهداف رفتاری**

با توجه به این که خودآزمایی های فصل بر اساس اهداف رفتاری طراحی شده اند، جدول ارتباط تمارین کمک می کند هنرآموزان عزیز دریابند هر هدف رفتاری آموزشی در فصل، به وسیله ی چه نوع سوالی مورد ارزشیابی قرار گرفته است.

- **راهنمای حل تمرین فصل**

برخی از تمرین ها به دلیل خلاصه شدن صورت سوال یا دلایل دیگر ممکن است نیاز به شرح بیشتر داشته باشد. این گونه تمرین ها برای هر فصل یا حل شده اند یا راهنمایی برای حل دارند.

بخش سوم: راهنمای فصل به فصل کتاب درسی

کتاب درسی شامل اجزای مرتبط به هم است و این ارتباط از یک نظم منطقی برخوردار است. یادگیری مبتنی بر دلیل و عملی مواد آموزشی هر فصل بر یادگیری مواد آموزشی سایر فصل ها تاثیر گذار است. به این ترتیب می باید ضمن پوشش کامل مطالب، ارتباط و چرایی تدوین هر مورد آموزشی به دقت تبیین و تعیین شود.

تدریس جزیره ای و فاقد ارتباط، انباشتی از موارد را در ذهن مخاطب فرو می ریزد و در انتها نیز نتیجه ی قابل قبول به دست نمی آید. به همین خاطر در کتاب حاضر سعی شده است برای هر فصل متناظر با کتاب درسی، به صورت متناظر راهنمای آموزشی با تکیه بر تعلیل و تحلیل تهیه و ارایه شود تا یک نگاه فرایندی در آموزش فراهم شود و مخاطب با آشنایی بهتر از هر فصل در نهایت به یک یادگیری منسجم و هدف مند نایل شود.

فصل ۱

آشنایی با مفاهیم پایه ای و پردازش داده ها

اهداف رفتاری

مفهوم داده، اطلاعات و پردازش را بیان کند و آنها را به کار بندد.

تعریف برنامه، برنامه نویسی، مترجم و انواع زبان های برنامه نویسی را از نظر سطح نزدیکی به زبان محاوره ای بیان کند.

در یک برنامه، ورودی، خروجی و پردازش را معین کند.

هدف از برنامه نویسی را بیان نماید.

آشنایی با مفاهیم پایه ای و پردازش داده ها

کلیدواژه ها: مراحل طراحی نرم افزار، داده، اطلاعات، ورودی، خروجی

مقدمه

ما در زندگی روزمره خود برای مسائل مختلف اقدام به تولید راه حل می نماییم.در واقع پی ریزی و ایجاد نقشه ای درست و اصولی، برای رسیدن به هدف (حل مساله) گام اساسی و اولیه در این زمینه است. در ایجاد یک نرم افزار نیز باید با آشنایی کامل از مفاهیم اولیه به طراحی راه حل و اقدامات لازم دیگر در جهت انجام آن برآمد. هرگونه اقدامی تا به ثمر رسیدن نتیجه (تولید نرم افزار) بایدطی یک فرایند منظم و مشخص باشد.

۱-۱- مفاهیم پایه

به منظور کاربرد کامپیوتر در شاخه های مختلف باید نرم‌افزار متناسب با آن کاربرد را تهیه کرد. برای درک بهتر طراحی نرم افزار ،بهتر است ابتدا به مرور مفاهیم پایه ای (همچون داده،اطلاعات و...) بپردازیم و سپس مراحل طراحی نرم افزار را بررسی نماییم

۱-۱-۱-داده[۱]

تمامی مقادیری چون کلمات، اعداد و نمادها که به نظر بی‌مفهوم می‌رسند، می‌توانند نمونه‌ای از داده‌ها باشند. داده‌ها در اصل حقایقی از مشاهدات و تحقیقاتی هستند که برای امر خاصی جمع آوری شده‌اند. به بیان دیگر می‌توان داده‌ها را مواد اولیه ای در نظر گرفت که در یک یا چند عملیات(پردازش) به کار گرفته می شوند
نمرات آزمون‌های گرفته شده در یک کلاس، عددی که یک دماسنج نمایش می‌دهد، تعداد پرسنل یک شرکت و یا تعداد جمعیت یک شهر یا کشور، نمونه‌های از داده ها می‌باشند.

باید توجه داشت داده های یک سیستم یا یک عملیات می توانند حاصل پردازش سیستم دیگر باشند.به عنوان مثال نمره های دانش آموزان که می تواند داده ای برای یک برنامه محسوب شوند برای هر دانش آموز از حاصل جمع نمره های سوال های امتحان آن دانش آموز به دست می آید.

---

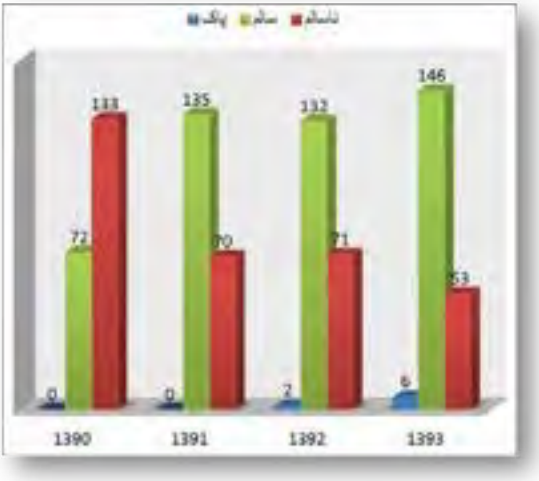
۱ Data

نام خانوادگی و نام	نوبت اول		نمره صفر		نمره پایانی	
	صفر	پایانی	به حروف	به عدد	به حروف	به عدد
حسن	۱۶		شانزده			
مهدی	۱۸		هجده			
ابراهیم	۱۴		چهارده			
حسن	۲۰		بیست			
[illegible]	۱۲		دوازده			
[illegible]	۱۶		شانزده			
[illegible]	۱۹		نوزده			
[illegible]	۱۰		هفت			
[illegible]	۱۳		سیزده			
[illegible]	۱۵		پانزده			
[illegible]	۱۸		هجده			
[illegible]	۱۳		سیزده			
[illegible]	۱۹		نوزده			
[illegible]	۱۶		شانزده			
[illegible]	۱۵		پانزده			
[illegible]	۲۰		بیست			

شکل ۱-۱ : نمونه‌های از داده‌ها

۱-۱-۲- اطلاعات[۱]

داده‌های جمع آوری شده با هدف مشخص پردازش شده تا بیانگر مفهوم یا حقایقی قابل درک و تفسیر باشند. به عبارتی دیگر اطلاعات، حاصل پردازش روی داده ها می باشند. مثلا با در نظر گرفتن نمرات آزمون درسی در یک کلاس می‌توان به پیشرفت آن کلاس پی‌برد. و یا با بررسی اعداد نمایش داده شده در طی روز، ماه و یا سال توسط دماسنج روند تغییرات زمین را بررسی نمود و همچنین با بررسی داده‌های ارسالی از ایستگاه‌های سنجش آلودگی هوا در شهر، تعداد روزهای پاک، سالم و ناسالم و روند رشد یا کاهش آلودگی را طی چند سال گذشته در پایتخت کشور عزیزمان ایران، مورد بررسی قرار داد.



شکل ۱-۲ : نمودار اطلاعات بدست آمده از داده‌های ورودی

۱ Information

۱-۱-۳- پردازش[۱]

عملیاتی که بر روی داده‌ها صورت می‌گیرد تا به اطلاعات تبدیل شوند را پردازش گویند. مثلا معدل گیری نمرات یک فرد یا یک کلاس، برای بررسی عملکرد آن فرد یا کلاس در طی دوره مشخص از دوران تحصیل، محاسبه تعداد ساعات حضور یک کارمند جهت پرداخت حقوق و دستمزد آن فرد، نمونه‌ای از عملیاتی است که بر روی داده ها انجام می‌شود تا به اطلاعات قابل استفاده تبدیل گردند.

۱-۱-۴- سیستم(سامانه)[۲]

به مجموعه اجزایی که در راستای یک هدف با همدیگر همکاری می کنند سیستم گفته می شود. سیستم ها دارای ساختار و رفتار مشخص هستند و میان اجزای ساختار آنها ارتباط برقراراست. به عنوان نمونه، می توان از سیستم نرم افزاری اطلاعات خودرو ها و از سیستم خودرو (به عنوان سیستم سخت افزاری) نام برد.

۱-۱-۵- ورودی[۳]

به آنچه که به یک سیستم وارد می شود تا طی عملیاتی برای به دست آمدن نتیجه استفاده شود ورودی اطلاق می شود. به عنوان مثال ورودی سیستم بینایی نور است.
در یک سیستم نرم افزاری نیز ورودی ها که نقش داده های آن را ایفا می کنند در پردازش ها مورد استفاده قرار می گیرند.به عنوان مثال در سیستم اطلاعات دانش آموزان، هر یک از مشخصات یک دانش آموز باید در سیستم وارد شود.این ورودی ها برای ثبت و انجام پردازش ها مورد استفاده قرار می گیرند.

۱-۱-۶- خروجی[۴]

خروجی سیستم، حاصل پردازش یک سیستم است که برای استفاده ، به محیط بیرونی آن سیستم انتقال پیدا می کند.
اگر بخواهیم اطلاعات موجود در یک سیستم را نمایش بدهیم باید آن را به عنوان خروجی سیستم مشخص نموده و دستورات لازم را صادر نماییم.
تعیین ورودی و خروجی بستگی به سیستم مورد نظر دارد. به این معنی که با تعیین یک سیستم مشخص می شود آیا داده ای ورودی است یا اطلاعاتی خروجی می باشد یا خیر!

۱-۱-۷- برنامه[۵]

برنامه، مجموعه دستورالعمل‌هایی است، که برای برآوردن یک هدف و کار خاص نوشته شده اند . در مثال‌های قبل برای بدست آوردن معدل فرد یا کلاس و همچنین پرداخت حقوق پرسنل شرکت مورد نظر، از روشهای خاص و متفاوت باید استفاده شود، یعنی باید

---

۱ processing
۲ system
۳ Input
٤ output
٥ program

نحوه محاسبات به کامپیوتر، طبق برنامه‌های متفاوت داده شود تا کامپیوتر برای هر نوع از محاسبات روشی را در نظر بگیرد و خروجی مطلوب حاصل شود.

۱-۱-۸-برنامه نویس[۱]

به اشخاصی که دستورالعمل‌های انجام کاری را، پس از تحلیل مساله و با الگوریتم منطقی و به زبان صوری برای کامپیوتر می‌نویسند، برنامه‌نویس گویند. برنامه‌ها بسته به نوع کاربرد آن می‌توانند ساده و یا خیلی پیچیده باشند، به گونه‌ای که نوشتن یک برنامه به کمک یک فرد شاید کمتر از یک ساعت طول بکشد و درمقابل برنامه دیگری، با تیمی از برنامه نویسان به زمانی بیش از یک سال نیاز داشته باشد.

برای درک بهتر مفاهیم داده، اطلاعات و پردازش به مثال سیستم حضور و غیاب با اثر انگشت توجه کنید:

- **داده:** تصویر اثر انگشت
- **پردازش:** دریافت تصویر و استخراج ویژگی ها و تطبیق ویژگی ها با بانک اطلاعات و در صورت تطبیق ثبت تاریخ ورود یا خروج.
- **اطلاعات:** ثبت تاریخ ورود و یا تاریخ خروج

۱-۲-مراحل طراحی نرم افزار

طراحی نرم افزار به شکل ساخت یافته دارای مراحلی می باشد که به صورت زیر می توان آن را دسته بندی نمود:

۱) **شناخت و حل مساله**
۲) **پیاده سازی و اجرا**
۳) **مستند سازی و نگهداری**

۱-۲-۱- شناخت و حل مساله

اولین و مهمترین قدم برای شروع برنامه نویسی شناخت مسئله است. در این مرحله صورت مساله را به خوبی بررسی کرده و نیاز ها و شرایط آن را شناسایی می کنیم. سپس در مرحله بعد اقدام به ایجاد راه حل می کنیم. این راه حل ها همان الگوریتم[۲]های حل مساله می باشند. هر کدام از راه حل ها به صورت دستی آزمایش و بررسی می شوند. سپس بهترین راه حل را انتخاب می کنیم. پس می توان این گام را به مراحل زیر تقسیم نمود:

الف- شناخت دقیق مساله

---

۱ programmer
۲ روش حل جزء به جزء و دقیق مساله

ب- طراحی راه حل ها

ج- بررسی و آزمایش دستی راه حل ها

د- انتخاب بهترین راه حل (الگوریتم بهینه)

به عنوان مثال برای نوشتن یک برنامه، جهت کنترل یک روبات مسیریاب، برنامه‌نویس باید با اصول کارکرد روبات کاملاً آشنا باشد؛ باید بداند که چگونه یک روبات مسیر خود را شناسایی می‌کند تا بتواند در آن مسیر به حرکت خود ادامه دهد. به بیان دیگر می‌توان گفت که برنامه‌نویس باید ورودی‌ها[1] و خروجی‌های روبات را شناسایی کند. همچنین لازم است شرایط مسیر و دیگر خواسته ها را بررسی کند.سپس با استفاده از دانسته ها و تحلیل بررسی ها اقدام به ایجاد راه حل کرده و بهترین راه حل را انتخاب نماید.

شکل ۱-۳ : روبات مسیر یاب ساده

۱-۲-۲- پیاده سازی و اجرا

بعد از انتخاب الگوریتم بهینه،باید مراحل الگوریتم با کدهای برنامه نویسی پیاده سازی شوند. در ابتدا باید نوع زبان برنامه نویسی انتخاب شود.بنابراین باید آگاهی کافی از انواع زبانهای برنامه نویسی وجود داشته باشد:

۱-۲-۲-۱- انواع زبانهای برنامه نویسی

برنامه‌های قابل اجرا در کامپیوتر، باید به زبان ماشین (صفر و یک) باشند، لذا نوشتن آن برای یک برنامه نویس کاری بسیار دشوار است. پس سعی می‌شود زبان‌های برنامه نویسی به زبان محاوره‌ای انسان نزدیک باشند از این جهت می‌توان زبان‌های برنامه نویسی را در سه سطح پایین، بالا و میانی تقسیم بندی نمود.

---

۱ سنسورهای فتودیود یا فتوترانزیستور در ربات‌های مسیریاب، اختلاف رنگ سیاه و سفید مسیر حرکت را شناسایی و سیگنال مناسب را به ورودی برد کنترلر ارسال می کنند.

در تقسیم بندی سطوح زبان‌های برنامه‌نویسی، زبان ماشین به زبان سطح پایین معروف است. این بدین معنا نیست که ارزش کمتری نسبت به زبانهای سطوح دیگر دارد بلکه نشانگر عدم وجود انتزاع بین زبان ماشین و دستورات قابل فهم پردازنده است. برنامه‌نویسی در این سطح دشوار است چون برنامه‌نویس باید شناخت کاملی از ساختار پردازنده داشته باشد تا بتواند از کدهای قابل فهم پردازنده برای برنامه‌نویسی استفاده کند. برنامه‌نویس در این سطح به راحتی می‌تواند محتوای ثبات‌ها را مقداردهی و یا فراخوانی کند. اگر برنامه‌ها، در این سطح، اصولی نوشته شده باشند، نسبت به برنامه نوشته شده در سطوح بالاتر خیلی سریعتر و با میزان حافظه کمتری اجرا خواهند شد. میزان حافظه ی اشغالی در برنامه ای که با زبان سطح پایین نوشته شود به مراتب کمتر از زبان سطح میانی و زبان سطح بالاست. به دلیل اینکه کدهای نوشته شده در زبان سطح بالا و میانی در نهایت با استفاده از یک مترجم باید به زبان سطح پایین ترجمه شوند، در نتیجه کدهایی مازاد بر کدهای ضروری نیز افزوده شده و زمان اجرای برنامه و زمان پاسخگویی افزایش خواهد یافت.

در برنامه‌نویسی سطح بالا دستورات شباهت زیادی به کلمات زبان محاوره‌ای دارند. لذا برنامه‌نویسی در این سطح آسانتر خواهدبود. برنامه‌نویس در این سطح، نیازی به درک جزئیات عملکرد داخلی پردازنده نداشته و به جای استفاده از ثبات‌ها و آدرس‌دهی حافظه کامپیوتر، با متغیرها کار می‌کند. محیط برنامه‌نویسی سطح بالا، گرافیکی و کاربرپسند است. دستورات استفاده شده به زبان انگلیسی نزدیکتر بوده که خود باعث جذابیت بیشتر است و فراگیری آن را نیز ساده تر می‌کند.

زبان برنامه نویسی سطح میانی همانطور که از نامش مشخص است مابین سطوح یاد شده قرار دارد. بدین معنا که قابلیت‌های آدرس‌دهی و دسترسی مستقیم به بیت‌ها و بایت‌های کامپیوتر را به برنامه‌نویس می‌دهد و از طرف دیگر خوانایی زبان برنامه نویسی سطح بالا را دارد. زبان برنامه‌نویسی C یکی از این زبان‌هاست. در این زبان هیچ محدویتی برای برنامه نویس وجود ندارد، لذا برنامه‌نویس هر آنچه فکر کند، می‌تواند با دستورات این زبان پیاده سازی کند. بیشتر برنامه‌های سیستمی، درایور سخت افزارها، برنامه بوت سیستم و برنامه های تخصصی کنترل پورت‌ها، نمونه برنامه‌هایی هستند که با زبان سطح میانی نوشته می‌شوند.

علاوه بر تفاوت سطح، هر زبان برنامه نویسی دارای ویژگی های خاص خود از لحاظ ساختار،نوع اجرا،نوع دستورات و بهینه بودن برای نوعی از کاربرد می باشد[1] برنامه نویس با توجه به برنامه ای که می خواهد بنویسد ابتدا باید تشخیص دهد کدام سطح از زبان و کدام زبان برنامه نویسی برای کار وی مناسب خواهد بود. جهت مقایسه زبانهای سطح بالا، میانی و پایین برنامه نمایش یک عبارت ، به سه زبان C++، C# و اسمبلی در بخش "مقایسه یک برنامه در زبان های مختلف"، ارایه شده است.

۱ برای دیدن مقایسه برخی زبان ها، می توانید به آدرس http://www.cprogramming.com/langs.html مراجعه فرمایید.

۱-۲-۳-مستندسازی و نگهداری

مستند سازی یک نرم افزار از مرحله تشخیص نیاز ها تا اتمام برنامه نویسی و انتشار برنامه جزو کارها و وظایف مهم برنامه نویسان می باشد. اهمیت این کار هم به دلیل نقش آن در توسعه نرم افزار و هم استفاده در مراجعات بعدی به کد های نرم افزار است.

هم چنین پس از اتمام کار برنامه نویسی ،نرم افزار ایجاد شده نیاز به نگهداری دارد. نگهداری نرم افزار به معنی رفع خطاهای پیش آمده بعد از انتشار نرم افزار و بهبود و به روز رسانی آن است.

۱-۳- مقایسه یک برنامه در زبان های مختلف

همان طور که قبلا اشاره شد دومین مرحله طراحی نرم افزار پیاده سازی و اجرای آن است. میزان کدنویسی که برای پیاده سازی انجام میشود بستگی به سطح زبان برنامه نویسی مورد استفاده دارد. در ادامه خواهید دید که یک برنامه یکسان، در زبانهای مختلف با استفاده از چه تعداد دستور نوشته میشود.

در این بخش برنامه نمایش عبارت "hello world!" در سه زبان اسمبلی،C++ و C# را مرور می کنیم.

زبان سطح پایین : زبان اسمبلی (برای سیستم عامل Dos و پردازنده X۸۶ اینتل)

```
.model tiny
.code
org 100h
main  proc
mov    ah,9
mov    dx,offset hello_message
int    21h
retn
hello_message db 'hello world!$'
main  endp
end   main
```

زبان سطح میانی : زبان++C

```
int main()
{
   std::cout << “hello  world!”;
}
```

زبان سطح بالا: زبان #C

```
using System;
class Program
{
   public static void Main(string[] args)
   {
      Console.WriteLine(“hello  world!”);
   }
}
```

واژگان

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	Application Software	نرم افزار کاربردی
۲	Assembly Language	زبان اسمبلی
۳	Compiler	مترجم
۴	Data	داده
۵	Feedback	بازخورد
۶	High Level Language	زبان سطح بالا
۷	Information	اطلاعات
۸	Installation	نصب
۹	Low Level Language	زبان سطح پایین
۱۰	Machine Language	زبان ماشین
۱۱	Medium Level Language	زبان سطح میانی
۱۲	Problem Solving	حل مساله
۱۳	Process	پردازش
۱۴	Program	برنامه

پرسش و پاسخ

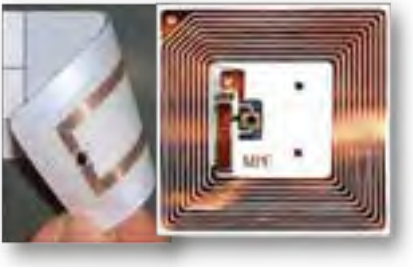
۱- چه نیازی به بیان توضیح مفاهیم داده، پردازش، ورودی و خروجی و .... در فصل اول است؟

برای حل مساله و نوشتن یک برنامه باید هنرجویان دریابند که کدام ورودی طی چه پردازشی باید به خروجی مورد نظر تبدیل شود. لذا اولین گام در نوشتن یک برنامه تشخیص ورودی، پردازش و خروجی آن میباشد که در بند سوم هدفهای رفتاری فصل اول آمده است و خودآزمایی های یکم، چهارم، پنجم و ششم در راستای همین هدف هستند.

۲- RFID [۱]چیست ؟

ضرورت جمع آوری دادهها در عرصههای خدماتی، صنعتی و علمی امروزه با افزایش دادهها بسیار احساس میشود لذا این امر اگر بدون نیاز انسان و به صورت شناسایی خودکار[۲] صورت گیرد خطاهای انسانی حذف و باعث افزایش سرعت، کارآیی و سرویس دهی بهتر خواهد شد.

به مجموعه ای از فنآوری ها که در آن برای شناسایی خودکار افراد یا اشیاء از امواج رادیویی استفاده میکند، فناوری RFID گفته می شود که میتواند تلفیقی از کدخوان، تراشه برچسب، آنتن و فرستنده باشد. میدان مغناطیسی تولید شده به وسیله کدخوان باعث میشود ولتاژ مورد نیاز کارکرد تراشه برچسب RFID تامین شود و تراشه به کمک آنتن تعبیه شده، اطلاعات لازم را برای کدخوان ارسال کند. کدخوان امواج رادیویی برگردانده شده از تگ RFID را به دادههای دیجیتال تبدیل می-نماید تا در ادامه، امکان ارسال داده برای کامپیوتر و پردازش آن فراهم شود. کارت مترو و اتوبوس شهری، کارتهای ورود و خروج برای کارمندان و دانشجویان، برچسبهای کالا در فروشگاهها، کتابفروشیها و جهت انبارداری کالا، نمونههای از تگهای RFID میباشند که میتوان به آنها اشاره نمود.



شکل۱-۴ : نمونههایی از برچسب RFID

۱ radio frequency identification

۲ Auto ID

۳- ECU[۱] چیست ؟

برای بدست آوردن بهترین راندمان در یک اتومبیل باید ترکیب سوخت و هوا ورودی موتور به دقت تنظیم گردد این امر در اتومبیل قدیمی بر عهده قسمت مکانیکی بنام کاربراتور بوده است ولی همان طور که می‌دانید قسمت‌های مکانیکی بعد از گذشت مدتی نیاز به تعویض و یا تنظیمات مجدد خواهند داشت تا دقت قبلی خود را حفظ کنند. با توجه به پیشرفت الکترونیک و کامپیوتر این امر را به دستگاهی به نام ECU سپردند که دقت و کارایی بالاتری از خود نشان داده است.

ECU یک کامپیوتر کوچک است که با استفاده از سنسورهای مختلف همچون سنسور دریچه گاز، سنسور دمای آب ، سنسور دور موتور و... که در موتور اتومبیل قرار داده شده‌اند، داده‌های ورودی خود را برای تحلیل عملکرد موتور دریافت می کند و طبق برنامه‌ای که برای پردازنده آن نوشته شده است میزان ترکیب سوخت و هوا را با تنظیم مدت پاشیده شدن سوخت، مقدار هوای ورودی و زمان مناسب برای جرقه‌زنی را فراهم می‌سازد تا شرایط مناسب برای کار کرد موتور ایجاد شود و بیشترین راندمان با توجه به مقدار سوخت مصرفی و کمترین آلایندگی محیط زیست حاصل شود.

شکل ۱-۵ : ECU

۱ Electronic Control Unit

واحد کار

واحد کار شماره: ۱-۱	فصل ۱ - صفحه ۱تا ۷	مدت آموزش: ۳۶۰ دقیقه	روش تدریس پیشنهادی: نمایش- پرسش و پاسخ.
عنوان: برنامه تهیه خوشمزه ترین سالاد الویه سیستمی دنیا		هدف کلی: تعریف، تشخیص و بکارگیری سه مفهوم اصلی داده ورودی، عملیات پردازش، و خروجی اطلاعات در برنامه نویسی و مسایل روزمره.	
اهداف رفتاری: انتظار می رود دانش آموز ۱. دلیل نوشتن برنامه را بگوید ۲. مفهوم داده، اطلاعات و پردازش را بیان، تشخیص و در مساله بکار گیرد. ۳. در مسایل روزمره این موارد را تشخیص داده و بررسی نماید		اهداف پنهان درس: تزریق تفکر الگوریتمی برای به دست آوردن نتیجه هدف ایده آل: به کار گیری سه فاکتور مهم ورودی، پردازش و اطلاعات در مسایل روزمره، تجزیه و تحلیل مساله، تفکر در زندگی به روش سیستمی، و بازخورد و مرور سه مرحله در صورت موفق نبودن در فعالیتها و اهداف در زندگی روزمره.	

فرایند تدریس:

در ابتدای کار، بدون نام بردن از برنامه نویسی و مطالب درسی، هدف از این جلسه که همان تهیه سالاد باشد را بازگو می کنیم. از دانش آموزان می پرسیم که مواد اولیه سالاد چیست؟ با در اختیار قرار دادن مواد اولیه، می پرسیم آیا آنچه در دست دارند را می توانند به عنوان سالاد الویه مصرف نمایند؟ منتظر می شویم تا به لزوم انجام عملیات، از طرف دانش آموزان اشاره شود بعد از انجام عملیات تهیه سالاد، نتیجه را در ظرف یا قالب های خاص می ریزیم و قبل از مصرف از دانش آموزان می خواهیم که بگویند چرا ما برای تهیه غذا و چیزهای دیگری که می توانیم خودمان تهیه کنیم آن را از بازار نمی خریم؟ کدام مورد بهتر است: تولید سفارشی یا تولید معمولی و با ویژگی های عمومی؟ در انتها با تشریح ویژگی های ورودی، داده، عملیات (پردازش) و خروجی، اطلاعات سوالات زیر را از دانش آموزان می پرسیم:
در فرایند تولید سالاد الویه،کدام عوامل نقش های داده، پردازش و اطلاعات را می توانند بر عهده بگیرند؟ رابطه اطلاعات و داده با توجه به رابطه مواد اولیه و سالاد چیست؟ اگر می خواستیم با همین مواد، غذای دیگری بسازیم از همین دستورالعمل ها استفاده می کردیم؟ اگر برای تهیه سالاد، از مواد دیگری علاوه بر مواد حاضر استفاده می کردیم چطور؟ بر این اساس، دستورالعمل ها بر چه اساس طراحی می شوند؟ آیا می توانیم خروجی را به شکل دلخواه خود نمایش دهیم؟ ما می توانیم سالاد را برای مصرف آن در در یخچال نگهداری کنیم.
آیا همین کار را برای اطلاعات می توانیم انجام دهیم؟ آیا محصول تولیدی ما می تواند به عنوان یک ماده اولیه در جایی دیگر به کار برده شود؟ اطلاعات چطور؟

واحد کار شماره : ۱-۱	فصل ۱ - صفحه ۱تا ۷

توصیه های اجرایی:

- هدف آموزش مفاهیم اولیه و نحوه طراحی دستورالعمل ها برای رسیدن به نتیجه است بنابراین باید ذهن دانش آموزان طوری هدایت شود که به این مسائل فکر کنند. امکان دارد تمام ذهن آنها فقط معطوف به ظاهر کار شود که باید تدبیر لازم را برای هدایت، اتخاذ کرد.
- در صورت موفقیت آمیز بودن کار، سه مرحله فرایند به کمک دانش آموزان مرور گردد و اشکال یابی یا بهینه شود.
- اگر چندین ایده و راه پیشنهاد شد، با نوشتن همه آنها، بهترین و بهینه ترین راه با همکاری خود آنها، انتخاب شود.

ارزشیابی از انتظارات:

۱. داده، به عنوان مواد اولیه عملیات مصرف می شود	ص	غ
۲. اطلاعات، داده هایی هستند که در خروجی نمایش داده می شوند	ص	غ
۳. پردازش همان عملیات می باشد.	ص	غ
۴. دستورالعمل ها بر اساس خروجی یا هدف مورد نیاز و ورودی ها طراحی می گردند	ص	غ
۵. داده و اطلاعات در پردازش مصرف می گردند.	ص	غ
۶. اطلاعات نمی توانند به عنوان داده یک پردازش دیگر مورد استفاده قرار گیرند	ص	غ

تکالیف و پروژه های پیشنهادی:

۱. دو مثال عینی دیگر که دارای ورودی، پردازش و خروجی باشد بیابید.
۲. اگر بخواهیم بلندقدترین دانش آموز مدرسه را بیابیم ورودی، پردازش و خروجی چه خواهد بود؟

نمونه سوال از فصل۱

درستی یا نادرستی هر عبارت را تعیین کنید. د برای درست و ن برای نادرست.

۱- داده ها، اطلاعات پردازش شده ی یک سیستم محسوب می شوند.

۲- زبان اسمبلی به زبان محاوره ای نزدیکتر است.

۳- مجموعه محاسبات و عملیاتی که بر روی داده ها صورت می گیرد را پردازش می نامند

در سوالات زیر گزینه صحیح را انتخاب کنید.

۴- در سیستم مدرسه، نمرات آزمون درس ریاضی برای محاسبه ی میانگین کلاس در درس ریاضی، نمونه ای از کدام مورد است؟

الف) information ب) Data ج) Process د) Flowchart

۵-فرض کنید برای محاسبه میانگین نمرات یک دانش آموز به جای عمل تقسیم برتعداد نمرات، عمل ضرب انجام شده باشد. در فرآیند رسیدن داده به اطلاعات کدام نکته رعایت نشده است؟

الف) صحت داده ها ب)درستی انجام محاسبات ج)روش انجام پردازش

۶-معلمی نمرات درس برنامه سازی دانش آموزان را با هم جمع می زند و میانگین آنها را محاسبه می کند و از روی میانگین تصمیم می گیرد که کلاس تقویتی برای دانش آموزان برگزار کند یا خیر. در این سیستم میانگین نمرات چه نقشی دارد؟

الف) داده ب) اطلاعات ج) الگوریتم د) پردازش

۷-مجموعه محاسبات و عملیاتی راکه بر روی داده ها انجام می گیرد را چه می نامند؟

الف)دانش ب)الگوریتم ج)فلوچارت د)پردازش

۸-اینکه در بین نمرات درس یک دانش آموز، نمره کمتر از صفر و بیشتر از ۲۰ نداشته باشیم، نشان دهنده کدام مورد از فرایند رسیدن از داده به اطلاعات است؟

الف)درستی انجام محاسبات ب)روش انجام پردازش ج)صحت داده ها

۹-نتایج حاصل از عملیات انجام شده بر روی داده، کدام گزینه است؟

الف)اطلاعات ب)پردازش ج)دانش د)برنامه

در جملات زیر، جاهای خالی را با عبارات مناسب پرکنید.

۱۰-مجموعه ای از مقادیر درباره یک موضوع یا شی که به صورت کمی یا کیفی نمایش داده می شود.....نام دارد

۱۱-نتیجه تفسیر و بررسی اطلاعات که می تواند مبنای تصمیم گیری برای انجام کاری شود ..... نام دارد.

۱۲-زبان قابل فهم کامپیوتر که دنباله ای از صفر و یک است را ..... می نامند.

۱۳- زبان برنامه نویسی که به زبان محاوره ای نزدیک است، زبان ..... نام دارد.

ارتباط تمارین فصل و اهداف رفتاری

ردیف	اهداف رفتاری فصل اول	خودآزمایی	تمرینات برنامه نویسی
۱.	مفهوم داده، اطلاعات و پردازش را بیان کند و انها را به کار بندد.	۱ و ۴	
۲.	تعریف برنامه، برنامه نویسی، مترجم و انواع زبان های برنامه نویسی را از نظر سطح نزدیکی به زبان محاوره ای بیان کند.	۲ و ۳	
۳.	در یک برنامه ورودی، خروجی و پردازش را معین کند.	۵ و ۶	
۴.	هدف از برنامه نویسی را بیان نماید.	۷	

آموزش فصل۱

صفحات۱-۷

مدرس:
خانم پونه پرنیان

شهر آق قلا
استان گلستان

زمستان ۱۳۹۳

فصل ۲

آشنایی با زبان C#

اهداف رفتاری

قالب کلی یک برنامه ساده در زبان C# را بیان کند.

نحوه تعریف یک کلاس و متد را در یک برنامه ساده بکار بندد.

با استفاده از یک ویرایشگر، برنامه ساده بنویسد و آن را ذخیره نماید.

برنامه ذخیره شده را با استفاده از مترجم در پنجره کنسول ترجمه کرده و سپس اجرا نماید.

با استفاده از متدهای مربوط به رنگ، تغییری در خروجی برنامه خود به وجود آورد.

آشنایی با زبان C#

کلید واژه: چارچوب نرم افزاری، .Net Framework، CLR، مترجم csc، متد Main

مقدمه

قبل از آنکه به سمت یادگیری یک زبان برویم خوبست که اطلاعاتی را درباره پایه شکل‌گیری و نحوه کارکرد آن فراگیریم. این موضوع درباره زبان‌هایی مانند C# که برپایه چارچوب نرم‌افزاری خاص استوار است، بیشتر صادق است. در این فصل با چارچوب دات نت آشنایی حاصل می‌شود و نحوه تبدیل کد به برنامه اجرایی و کارکرد این نرم‌افزارها مرور می‌شود.

۲-۱- چارچوب دات‌نت

چارچوب نرم‌افزاری، محیطی قابل استفاده برای سیستم یا زیرسیستم‌های نرم‌افزاری است. چارچوب‌های نرم‌افزاری می‌توانند طراحی‌های شی‌گرا داشته باشند و ممکن است شامل برنامه‌های پشتیبانی، کدهای کتابخانه‌ای، زبان‌های اسکریپتی و یا نرم‌افزار دیگری برای توسعه و گسترش باشند. چارچوب‌ها برای این طراحی شده‌اند که به برنامه‌نویسان و طراحان اجازه دهند تا زمان خود را بیشتر صرف نیازمندی‌های نرم‌افزار کنند.

دات نت فریم‌ورک (.NET Framework) یک بسته نرم‌افزاری تهیه شده به وسیله ی شرکت مایکروسافت است. ایجاد و توسعه این چارچوب در اواخر دهه ۱۹۹۰ و با نام «سرویس‌های نسل بعد ویندوز (NGWS[1])» شروع شد و در سال ۲۰۰۰ اولین نسخه آزمایشی این نرم‌افزار منتشر شد.

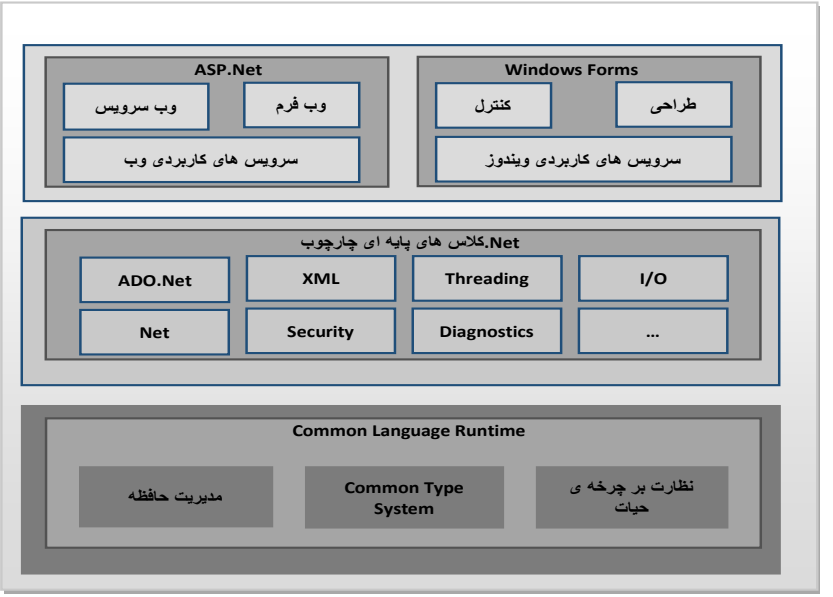
این چارچوب شامل راه‌حل بسیاری از نیازهای توسعه نرم‌افزاری، کتابخانه‌ی کلاس پایه[2] وهمچنین امکان توسعه و اجرای برنامه‌های تولید شده برای این فریم‌ورک است.

چارچوب دات نت، تمامی لایه‌های پیاده‌سازی نرم‌افزار را از سطح سیستم عامل به بالا، تحت پوشش قرارمی‌دهد. معماری فوق، امکان طراحی و پیاده‌سازی برنامه‌های مبتنی بر اینترنت، وب سرویس‌ها، برنامه‌نویسی شبکه،... و محیط‌های Desktop، را بسادگی فراهم و نیازهای هر گروه از نرم‌افزارهای فوق را بخوبی جواب می‌دهد. اجزای اصلی فریم‌ورک دات نت در شکل زیر نشان داده شده است.

---

۱ Next Generation Windows Services

۲ Base Class Library



شکل ۲-۱: اجزای اصلی فریم‌ورک دات نت

چارچوب دات نت از لایه پائین با عملیاتی نظیر مدیریت حافظه آغاز و به سمت بالا به منظور ارائه رابطهای برنامه‌ها و کاربران، دنبال می‌شود. ویژگی مهم Net.، پشتیبانی آن از چندین زبان برنامه‌نویسی متفاوت است. کد نوشته شده به وسیله یس این چارچوب و در زبان‌های مختلف آن، به یک زبان واسط(CIL[۳] یا MSIL[۴]) تبدیل می‌شود. CIL پایین‌ترین سطح زبان نزدیک به انسان است و طوری طراحی شده است که از چندین زبان برنامه‌نویسی پشتیبانی نماید. در حقیقت، هم اکنون ده‌ها زبان برنامه‌نویسی، مورد پشتیبانی و پذیرش CIL هستند. علاوه بر C#.net، دات نت شامل زبان‌هایی نظیر Visual Basic، Jscript، J# و C++ نیز است. برخی دیگر از زبان‌های برنامه‌سازی مهم که بوسیلهٔ CIL پشتیبانی می‌شوند عبارتند از : Cobol.Net ،Borland Delphi.Net، Fujitsu، Python.Net ،Perl.Net و ..

عامل اصلی نگهدارنده این زبان‌ها در کنار یکدیگر، سیستم نوع مشترک (CTS[5]) است. این سیستم، انواع مختلف (شامل class، structure، Interface و...) را تعریف، استفاده و مدیریت می‌کند.

فناوری دات نت بر روی تمامی ویرایش‌های سیستم‌عامل ویندوز مایکروسافت

۳ Common Intermediate Language

۴ MicroSoft Intermediate Language

۵ Common Type System

قابل اجراست و در سیستم‌عامل‌های دیگر از جمله لینوکس و مکینتاش[۶] نیز وارد شده است. کتابخانه‌های از پیش نوشته شده (FCL[7]) در این چارچوب که پایه‌های اشیاء و کلاس‌های اصلی این چارچوب را تشکیل می‌دهند، بخش وسیعی از نیازهای برنامه‌نویسی را تحت پوشش قرار می‌دهند. از این جمله می‌توان رابط گرافیکی کاربر، دسترسی به پایگاه‌های داده، رمزنگاری، برنامه‌های تحت‌وب، الگوریتم‌های کار با اعداد و ارتباطات شبکه‌ای را نام برد[۸].

همان طور که گفته شد کد تمام زبان‌ها به زبان واسط(CIL) ترجمه می‌شود و سپس برای اجرا به زبان ماشین ترجمه می‌گردد. بخش کامپایلر این چارچوب، یک مفسر درلحظه(JIT)[۹] است. .به این معنی که ترجمه از CIL به زبان ماشین **در زمان اجرا** صورت می‌پذیرد. این ترجمه به وسیله‌ی یک ماشین مجازی، فقط برای کدهایی انجام می‌شود که برنامه، در آن زمان به آن نیاز دارد. نام این ماشین مجازی CLR[۱۰] (زبان مشترک زمان اجرا) است. CLR محیطی را فراهم می‌کند، تا برنامه‌ها در آن اجرا شوند. کدهایی که تحت CLR و دات نت اجرا می‌شوند، کدهای مدیریت شده[۱۱] نامیده می‌شوند؛ زیرا CLR جنبه‌های مختلف نرم‌افزار را در زمان اجرا مدیریت می‌کند.

---

۶ Mono:

Mono نسخه متن باز برای net framework. است که برای زبان #C نوشته شده و روی برخی از سیستم‌عامل ها مثل لینوکس و macos .Android, BSD, iOS, Linux, OS X, Windows, Solaris and UNIX کار می‌کند.

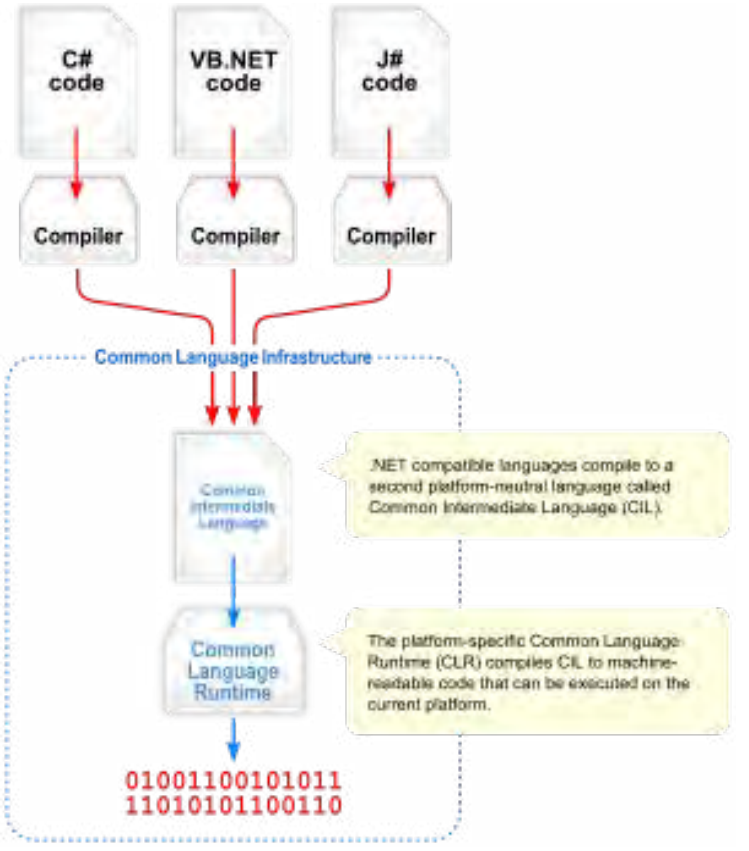
۷ Framework Class Library(FCL)

۸ NET Framework Class Library Overview

۹ Just in Time Compiler (JIT)

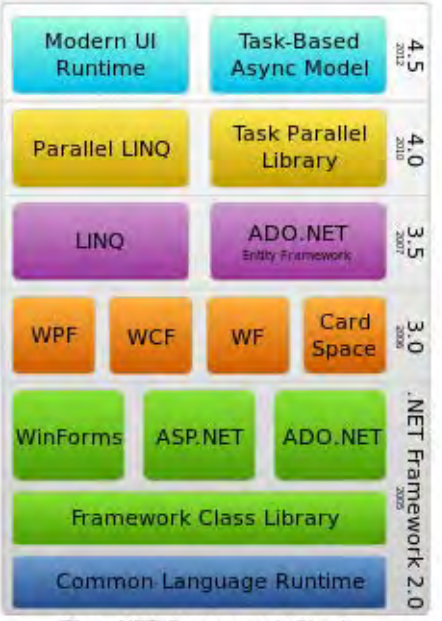
۱۰ Common Language Runtime

۱۱ managed code



شکل ۲-۲: مراحل ترجمه کد واجرا به وسیله‌ی CLR

در ژوئن سال ۲۰۰۰، شرکت مایکروسافت انتشار نسخه اول دات نت به همراه یک زبان برنامه‌نویسی جدید به نام سی‌شارپ را اعلام کرد. امکانات اضافه شده در نسخه‌های مختلف دات نت در شکل زیر نمایش داده شده است.



شکل ۲-۳: امکانات اضافه شده در نسخه‌های مختلف دات نت

زبان‌های سی‌شارپ و ویژوال بیسیک دات‌نت و ASP.NET همگی از چارچوب دات‌نت استفاده می‌کنند. دات نت باعث شد که قدرت برنامه‌نویسی در زبان‌های تحت این چارچوب یکسان شود و تفاوت فقط در نوع کد نویسی باقی بماند.

تغییر عمده‌ای که در برنامه‌نویسی با دات نت نسبت به گذشته قابل رؤیت است، روش اتصال به پایگاه‌های داده است، که علاوه بر روشهای قدیمی مانند DAO و ADO می‌توان از فناوری جدیدADO.NET نیز استفاده کرد.

۲-۲- آشنایی با زبان برنامه‌نویسی C#(سی شارپ)

زبانC# یک زبان شبه C [۱۲] است. زبان‌های شبه C از لحاظ نحوی بسیار شبیه زبان C هستند. زبان C، زبان برنامه‌نویسی است که به عنوان زبان hands-on توصیف شده است. یعنی برنامه‌نویس باید انجام همه چیز را قدم به قدم به برنامه بگوید. همچنین این زبان به شما تقریبا اجازه انجام هر چیزی را می‌دهد و شی‌گرایی را پشتیبانی نمی‌کند. بنابراین بدون کلاس است.

زبان C++، زبان توسعه‌یافته زبان C است. C++ به معنی افزایش یک سطح، به زبانC است. C++ از شی‌گرایی پشتیبانی می‌کند و اجازه کنترل بیشتر روی برنامه‌ها را می‌دهد و مانند زبانC زبان hands-on است و برنامه‌نویسی با

۱۲ C like program

جزئیـات بسـیار انجـام می‌شـود.

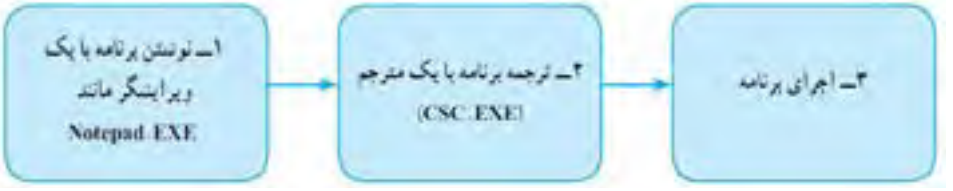
زبـان #C زبانـی کامـلا شـی‌گرا بـا کـدی شـبیه بـه سـبک C و C++ اسـت. #C آخریـن نسـخه از زبـان C و بسـیار مناسـب بـرای توسـعه برنامه‌هـای کاربـردی وب اسـت. ایـن زبـان به‌وسـیله‌ی شـرکت مایکروسـافت نوشـته شـده اسـت و ترکیبـی از قابلیت‌هـای خـوب زبان‌هـای C++ و **Java** اسـت. از سی‌شـارپ می‌تـوان بـرای نوشـتن برنامه‌هـای تحـت ویندوز، تحـت وب، وب سـرویس‌ها، برنامه‌هـای موبایـل و بازی‌هـا اسـتفاده کـرد. سی‌شـارپ از کتابخانـه کلاس دات نـت اسـتفاده می‌کنـد کـه شـامل مجموعـه بزرگـی از اجـزا از قبـل سـاخته شـده اسـت. ایـن اجـزا بـه سـاخت هـر چـه سـریع‌تر برنامه‌هـا کمـک می‌کننـد. سـاختار ایـن زبـان نسـبت بـه زبان‌هـای دیگـر بسـیار آسـان و قابـل فهـم اسـت.

بـرای اجـرای یـک برنامـه سی‌شـارپ ابتـدا بایـد دات نـت فریـم‌ورک نصـب شـود. البتـه بـا نصـب سیسـتم‌عامل ویندوز هفـت و بالاتـر بصـورت خـودکار دانـت نـت فریـم‌ورک روی سیسـتم نصـب می‌شـود و نیـاز بـه نصـب مجـدد آن نیسـت. در صـورت نیـاز بیشـتر می‌توانیـد آن را بـه روز رسـانی کنیـد. سی‌شـارپ یکـی از زبان‌هایـی اسـت کـه از تکنولوژی‌هـای دیگـر دات نـت ماننـد **ASP.NET**، **Silverlight**[۱۳] پشـتیبانی می‌کنـد. همچنیـن یـک محیـط توسـعه یکپارچـه بـا ابزارهـای مفیـد دارد و بـه شـما در کدنویسـی سی‌شـارپ کمـک می‌کنـد. یکـی از ویژگی‌هـای زبـان سی‌شـارپ پیشـتیبانی بـه روز بـه وسـیله‌ی شـرکت مایکروسـافت اسـت تـا قابلیت‌هـای جدیـد بـه آن اضافـه شـود.

۲-۳- اولین برنامه

بـرای شـروع برنامه‌نویسـی ابتـدا بایـد مقدمـات را فراهـم کـرد. مقدمـات مـورد نیـاز عبارتنـد از:

۱. ویرایشگر برای نوشتن برنامه
۲. مترجمی برای ترجمه برنامه به زبان ماشین
۳. اجرای فایل خروجی ساخته شده با مترجم



شکل۲-۴: مراحل برنامه‌نویسی، ترجمه و اجرا

۱۳. **Silverlight** : یک پلاگین آزاد است که در .NET FRAMEWORK طراحی شده و با بیشتر مرورگرها و سیستم عاملها سازگار است و سطح جدیدی از تعامل را برای کاربران در سطح وب ایجاد می‌کند.

در ادامه ساختار اصلی برنامه‌های C# و سپس مطالبی در مورد ویرایشگر Notepad++ و مترجم csc.exe آورده شده است.

۲-۴- ساختار اصلی برنامه‌های C#

قالب زیر ساده ترین شکل نگارش برنامه به زبان C# را نمایش می‌دهد.

```
Class نام کلاس
 {
  static void Main()
  {
دستورات برنامه
  }
 }
```

در این قالب کلمات static, class و void کلمات رزروشده در C# هستند که به آنها کلمات کلیدی گفته می‌شود و نباید از این کلمات در قسمت‌هایی استفاده شود که نیاز به نام‌گذاری به وسیله‌ی برنامه‌نویس دارد. نام کلاس یک نام دلخواه است و بهتر است برای خوانایی برنامه نامی متناسب با عملکرد کلاس باشد.

۲-۴-۱- کلمه کلیدی class

کلاس مفهوم خاص برنامه‌نویسی شی‌گرا است. در واقع قالب یا الگویی است که می‌تواند شامل داده‌های مرتبط با یک موجودیت و عملیات خاص روی آن داده‌ها باشد.

زبان C# شامل کلاس‌های از پیش تعریف شده‌ی زیادی است که کار برنامه‌نویس را ساده‌تر می‌کند. برنامه‌نویس کافیست که آنها را بشناسد تا بتواند به راحتی از آنها استفاده کند. یکی از کارهایی که تقریبا در اکثر برنامه‌ها به آن نیاز داریم، نمایش پیام ها روی صفحه نمایش است. انجام این عمل در کلاس پیاده‌سازی شده است. کلاس Console شامل دستوراتی برای عملیات ورودی و خروجی (دریافت اطلاعات از صفحه کلید و نمایش اطلاعات روی صفحه نمایش) است.

برای ایجاد کلاس باید برای آن نامی درنظر گرفت. این نام دلخواه است ولی نباید از کلمات کلیدی C# باشد. نام کلاس بلافاصله پس از کلمه کلیدی class قرار می‌گیرد و داده‌ها و عملیات روی آنها اعضای کلاس هستند و در داخل کلاس بین دو علامت } و { قرار می‌گیرند.

```
class نام دلخواه برای کلاس
 {
  داده‌ها و عملیات موردنظر
```

}

برای مثال کلاسی که کار آن محاسبه و چاپ سن افراد باشد، باید دارای داده تاریخ تولد یا سال تولد فرد و عملیات محاسبه سن و چاپ آن باشد. می‌توان آن را به صورت زیر با نام Age تعریف کرد.

```
class age
 {
   سال تولد(داده)
محاسبه سن(عملیات ۱)
نمایش سن روی صفحه نمایش(عملیات ۲)
 }
```

به عملیاتی که در کلاس تعریف می‌شود، متد می‌گوییم. در واقع متد مجموعه‌ای از دستورات برای انجام کار است. مانند عملیات محاسبه سن در مثال بالا. متدها مطابق عملی که انجام می‌دهند، نام‌گذاری می‌شوند. برای سال تولد نام Birthday، برای محاسبه سن، نام CalculateAge و برای چاپ سن نام PrintAge نام مناسبی است.

```
class age
 {
  birthday;
  CalculateAge();
  PrintAge();
 }
```

نام متدهای یک کلاس (داده و متدها و.....) می‌تواند با نام متدهای کلاس‌های دیگر یکسان باشد؛ بنابراین برای استفاده از متدهای کلاس باید قبل از نام متد، نام کلاس ذکر شده و این دو نام با نقطه از یکدیگر جدا شوند. این روش برای مشخص کردن کلاسی است که برنامه‌نویس قصد استفاده از متد آن را دارد.

نام متد . نام کلاس
Age.CalculateAge

۲-۴-۲- استفاده از متدهای آماده در C#

برای استفاده از متدهای تعریف شده در کلاس‌های آماده C# هم باید به روش ذکر شده در بالا عمل کرد.

نام متد.نام کلاس

کلاس Console دارای متد WriteLine برای نمایش اطلاعات و پیام روی صفحه نمایش است. پیام‌ها را باید بین دو علامت " قرار داد. برای مثال به منظور

نمایش پیام خوش آمدگویی Hello World! روی صفحه نمایش دستور زیر بکار می‌رود. این دستور پس از نمایش پیام، مکان نما را به ابتدای خط بعد منتقل می‌کند؛ بنابراین پیام بعدی در خط بعد نمایش داده می‌شود.

```
Console.WriteLine("Hello World!");
```

توجه: C# به حروف کوچک و بزرگ حساس[۱۴] است. دقت کنید که حرف C در Console و حروف W,L در WriteLine بزرگ و بقیه حروف کوچک هستند.

انتهای تمام دستورات C علامت ; قرار داده می‌شود که نشان دهنده پایان دستور است. به علت وجود این علامت در پایان هر دستور، می‌توان چندین دستور را در یک خط نوشت و یا یک دستور را در چندین خط نوشت.

```
Console.WriteLine("Hello "); Console.WriteLine("World!");
```

البته با این دو دستور در یک خط کلمه Hello و در خط بعد کلمه World! نوشته می‌شود.

و یا

```
Console.
WriteLine("  Hello World!");
```

در این حالت ما یک دستور داریم که در هنگام برنامه‌نویسی در دوخط نوشته شده است و در عمل با دستور زیر هیچ تفاوتی نخواهد داشت.

```
Console.
WriteLine("Hello World!");
```

برای سازماندهی کلاس‌های آماده در دات نت و همچنین سازماندهی کلاس‌ها در برنامه‌های بزرگ نیاز است که کلاس‌های مرتبط را در یک دسته قرار دهیم. به این منظور از فضای نام(namespace) استفاده می‌کنیم. همان طور که پوشه در کامپیوتر برای مرتب‌سازی و سازماندهی فایل‌ها مورد استفاده قرار می‌گیرد، فضای نام نیز در برنامه‌ها برای سازماندهی کلاس‌ها به کار می‌رود تا دسترسی به کلاس‌ها و متدها برای برنامه‌نویس آسان باشد. قالب زیر، الگوی برنامه را در صورت استفاده از namespace نمایش می‌دهد.

```
namespace نام دلخواه برای فضای نام۱
{
 class نام دلخواه برای کلاس۱
```

۱۴ Case sensetive

```
 {
        داده و عملیات کلاس۱
 }
 class  نام دلخواه برای کلاس۲
 {
        داده و عملیات کلاس ۲
 }
:
:
:
 class  نام دلخواه برای کلاس۳
 {
       static void Main()
         {
       دستورات برنامه
         }

 }
}
namespace  نام دلخواه برای فضای نام۲
{
 class  نام دلخواه برای کلاس۱
 {
        داده و عملیات کلاس۱
 }
 class  نام دلخواه برای کلاس۲
 {
        داده و عملیات کلاس ۲
 }
  :
:
:
}
```

در صورت سازماندهی کلاس‌ها با فضای نام، هنگام استفاده از متدها علاوه برتعیین نام کلاس باید فضای نامی که کلاس در آن قرار دارد را هم ذکر کنیم. به عنوان نمونه کلاس Console در فضای نام System قرار دارد؛ بنابراین متد WriteLine باید به صورت زیر در برنامه استفاده شود.

```
نام متد.نام کلاس.فضای نام

System.Console.WriteLine(“Hello World!”);
```

برای راحتی کار، برنامه‌نویس می‌تواند با استفاده از کلمه کلیدی using فضای نام را در اول برنامه معرفی کند. در این صورت هنگام استفاده از کلاس ذکر فضای نام لازم نیست.

```
using System;
 Console.WriteLine(“Hello World!”);
```

۲-۴-۳-متد Main

متد Main نقطه شروع اجرای برنامه است[۱۵](دقت کنید که حرف M در Main بزرگ است.). NET Framework. هنگام اجرای برنامه‌ای که به زبان C# و به صورت پروژه کنسول یا ویندوز نوشته شده، در فایل اجرایی به دنبال متدی به نام Main گشته، با پیدا کردن آن، کدهای داخل این متد را اجرا می‌کند. کدهایی که بین { و } نوشته می‌شوند بدنه متد Main هستند. برنامه با اجرای اولین دستور این متد شروع شده و دستورات بعدی آن به ترتیب اجرا می‌شوند. با رسیدن به آخرین خط متد و اجرای آن، از متد Main خارج شده، برنامه به اتمام می‌رسد؛ بنابراین برنامه نیاز به متد Main دارد و متد Main باید در کلاس تعریف شود.

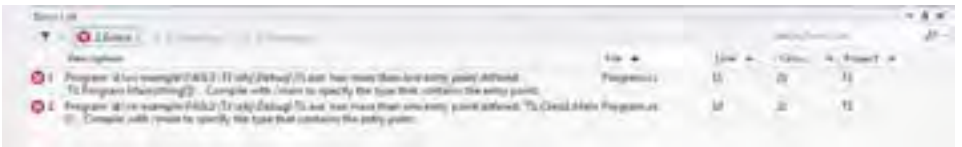
```
static void Main()
{
    دستور ۱;
    دستور ۲;
    دستور ۳;
    دستور ۴;
            :
            :
            :
}
```

پروژه‌ها می‌توانند دارای چندین کلاس باشند ولی تنها یکی از کلاس‌های دارای متد Main به عنوان مدخل ورود به پروژه است و اگر در پروژه بیش از یک متد Main وجود داشته باشد باید در هنگام ترجمه نقطه ورود به پروژه مشخص شود. به این منظور

۱۵ در برنامه هایی که به صورت console application یا windows application هستند متد Main نقطه شروع برنامه است و برنامه های از نوع web Application، Librariesو services نیاز به Main ندارند زیرا خروجی آنها فایل اجرایی نیست بلکه از خروجی آنها در پروژه‌های دیگر استفاده می‌شود.

کلاسی که متد Main آن نقطه ورود پروژه است، باید به عنوان کلاس پیش‌فرض برای راه‌اندازی پروژه، در تنظیمات properties پروژه مشخص شود وگرنه خطا رخ می‌دهد. به مثال زیر و خطای رخ داده توجه کنید.

```
using System;
namespace T1
{
 class Program
 {
  static void Main(string[] args)
    {
     Console.WriteLine("T1");
     Console.ReadKey();
  }
 }
 class Class1
 {
  static void Main()
  {
   Console.WriteLine("class1");
   Console.ReadKey();
  }
 }
}
```



شکل ۲-۵:خطای مشخص نبودن نقطه شروع برنامه به علت وجود دو متد Main در یک پروژه

۲-۴-۴- برنامه خوش آمدگویی

با توجه به مطالب ذکر شده در بالا، ساده‌ترین قالب برنامه شامل کلاس و متد Main است و نیازی به فضای نام ندارد؛ زیرا همانطورکه بیان شد فضای نام برای سازماندهی کلاس‌ها در پروژه‌های بزرگ به کار می‌رود و یک برنامه ساده چند خطی که فقط یک کلاس دارد نیاز به فضای نام ندارد.

برای اولین برنامه، ساده‌ترین قالب را که شامل کلاس و متد Main است به کار می‌بریم. نام کلاس را firstProgram قرار داده‌ایم و در متد Main دستور نمایش

پیـام «Welcome Students » را نوشـته‌ایم. در ایـن برنامـه از using اسـتفاده نشـده اسـت؛ بنابرایـن در فراخوانـی متـد WriteLine نـام کلاس و فضـای نـام متـد ذکـر شـده اسـت.

```
class firstProgram
{
    static void Main()
    {
        System.Console.WriteLine("Welcome Students.");
    }
}
```

با استفاده از using برنامه به صورت زیر تبدیل می‌شود.

```
using System;
class firstProgram
{
    static void Main()
    {
        Console.WriteLine("Welcome Students.");
    }
}
```

در ایـن حالـت بـرای فراخوانـی متـد WriteLine تنهـا ذکـر نـام کلاس آن کافیسـت و نیـاز بـه نوشـتن فضـای نـام آن نیسـت.

۲-۵- ویرایشگر برای نوشتن برنامه

معروف‌تریـن و متداول‌تریـن ویرایشـگر در سیسـتم‌عامل‌های وینـدوز برنامـه‌ی Notepad اسـت. ایـن برنامـه‌ی کـم حجـم و سـریع، در تمامی نسـخه‌های مایکروسـافت وینـدوز وجـود دارد و بـا ظاهـری سـاده امـکان ویرایـش متن‌هـا را بـه کاربـران داده اسـت. Notepad++ برنامـه‌ای رایـگان بـرای ویـراش کـد اصلـی[۱۶] برنامه‌هـا و جایگزینـی مناسـب بـرای Notepad وینـدوز اسـت. این‌نـرم افـزار، زبان‌هـای برنامه‌نویسـی متعـددی را پشـتیبانی می‌نمایـد. ویرایشـگر Notepad++ بـا توجـه بـه نـوع زبـان انتخـاب شـده کلمـات کلیـدی آن زبـان را بـه صـورت رنگـی نشـان می‌دهـد؛ بنابرایـن بـرای ویراسـتاری برنامه‌هـا مناسـب اسـت. توجـه داشـته باشـید کـه از نرم‌افـزار word بـرای ایـن منظـور اسـتفاده نکنیـد؛ زیـرا برخـلاف Notepad بـه انتهـای خطـوط و پارگراف‌هـا قالب‌بنـدی خاصـی را اضافـه می‌کنـد کـه هنـگام (کامپایـل) سـاخت فایـل

۱۶ source code

اجرایی تولید خطا می‌کند.

برنامه Notepad++ امکاناتی مانند مشخص نمودن syntaxها، تفکیک و قالب‌بندی کدها، ویرایش همزمان چند فایل، Drag & Drop متون، قابلیت جستجوی کامل و ... را دارد. قابلیت‌های نرم افزار Notepad++ عبارتند از :

- نمایش دستورهای زبان‌های برنامه‌نویسی مختلف به صورت رنگی
- پشتیبانی از زبان‌های برنامه‌نویسی مختلف مانند:

C, C#, C++, FORTRAN, HTML, Haskell, Java, javascript, Pascal, Perl, PHP, PostScript, Python, SQL, VB/VBScript, XML

- قابلیت معرفی کردن زبان جدید به وسیله‌ی کاربر، که به آن User Defined به معنی «تعریف شده توسط کاربر» می‌گویند. شما می‌توانید زبان برنامه‌نویسی جدید را با معرفی قوانین زبان مورد نظر به کمک Wizard (دستیار ویژه) (نشانه) به برنامه Notpad++ اضافه کنید، علامت‌های توضیحات[۱۷] و علامت‌های عملوندها و دیگر موارد مورد نیاز... در زبان جدید را مشخص کنید و به راحتی رنگ و قواعد را باب سلیقه خود ویرایش کنید. این اطلاعات در فایلی با پسوند xml. در محلی که برنامه Notepad++ نصب شده در مسیر Plugins\APIs ذخیره می‌شود.
- دارای Auto-completion است. بدین معنی که بعد از انتخاب کردن زبان برنامه‌نویسی خود و فشار دادن کلیدهای ترکیبی Ctrl+Space یک منو باز می‌شود که می‌توانید کلمات تعریف شده در آن زبان را به راحتی پیدا کنید و تایپ کنید.
- قابلیت (کشیدن و رها کردن) Drag & Drop، باز کردن اسناد به وسیله Drag & Drop به داخل نرم‌افزار یا انتقال اسناد در داخل محیط نرم‌افزار به وسیله Drag & Drop
- قابلیت باز کردن چندین فایل همزمان (قابلیت چند سندی[۱۸]) و امکان فراخوانی آنها با باز کردن مجدد نرم افزار

۱۷ Comment

۱۸ MDI

- پشتیبانی از قابلیت Bookmark(کاربر می‌تواند از قسمت کناری نرم‌افزار با فشردن کلیدهای Ctrl+F2، بوکمارک را فعال کند.)

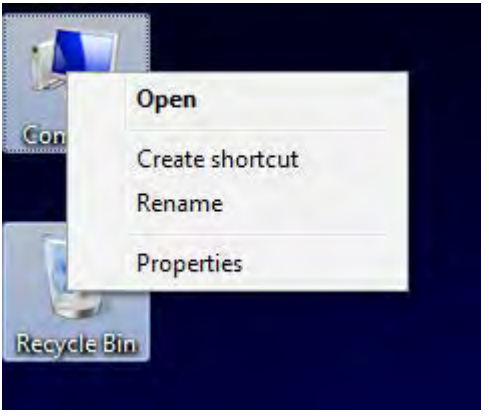
قبل از نوشتن برنامه در این محیط، زبان برنامه‌نویسی آن را روی C# تنظیم نمایید، در غیر این صورت در هنگام ترجمه و ساخت فایل اجرایی ممکن است به علت اضافه شدن قالب‌بندی زبانی غیر از زبان C# با مشکل مواجه شوید. بدین منظور از منوی language→ C→C# را انتخاب نمایید، سپس برنامه را نوشته وآن را با پسوند cs. در مسیر دلخواه ذخیره کنید.

۲-۶- مترجمی برای ترجمه برنامه

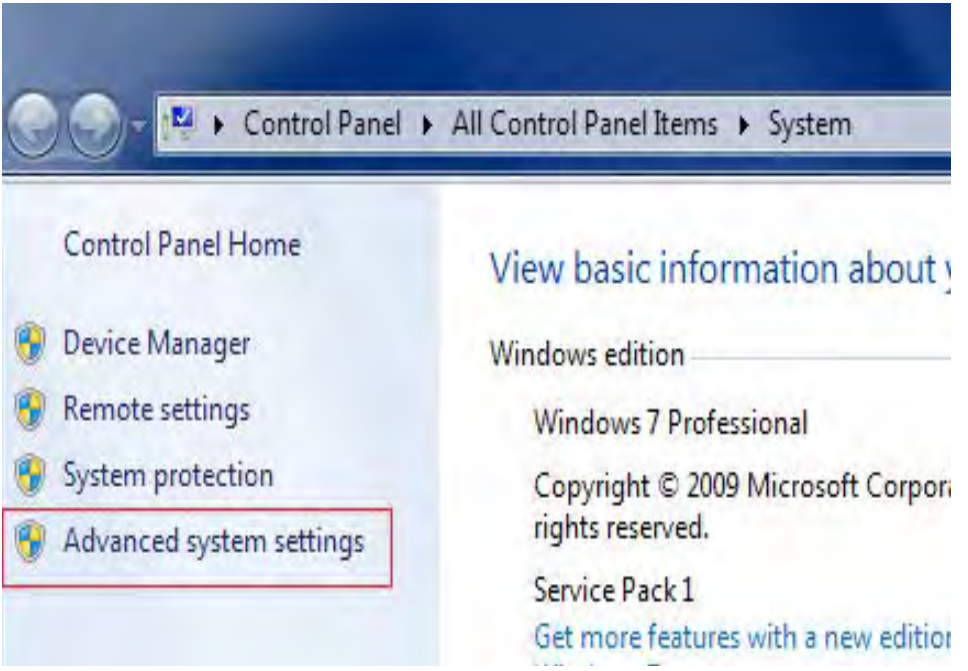
جهت ترجمه‌ی برنامه‌های نوشته شده به زبان C#.net از برنامه مترجم csc.exeدر محیط فرمان استفاده می‌کنیم؛ بنابراین باید مسیر فایل مترجم و فایل برنامه را به سیستم‌عامل اطلاع دهیم. البته باید یادآوری کنیم که با نصب نسخه ویندوز هفت و بالاتر در روی سیستم‌ها به صورت پیش فرض .NET Framework نصب می‌شود و برنامه مترجم csc.exe در مسیر C:\Windows\Microsoft.NET\Framework و داخل پوشه‌ای با نام vN قرار دارد. که N شما نسخه framework مانند v۲.۰.۵۰۷۲۷ است.

در صورت استفاده از خط فرمان استاندارد ویندوز (cmd) باید مسیر فایل csc.exe را به سیستم‌عامل اعلام کنیم. برای این منظور از دستور path استفاده می‌شود. برای راحتی تنظیم مسیر فوق به جای استفاده مستقیم از دستور path، از محیط رابط گرافیکی کاربر (GUI) استفاده می‌کنیم.

در win7 روی computer راست کلیک کرده و مطابق شکل‌های زیر به ترتیب عمل کنید.

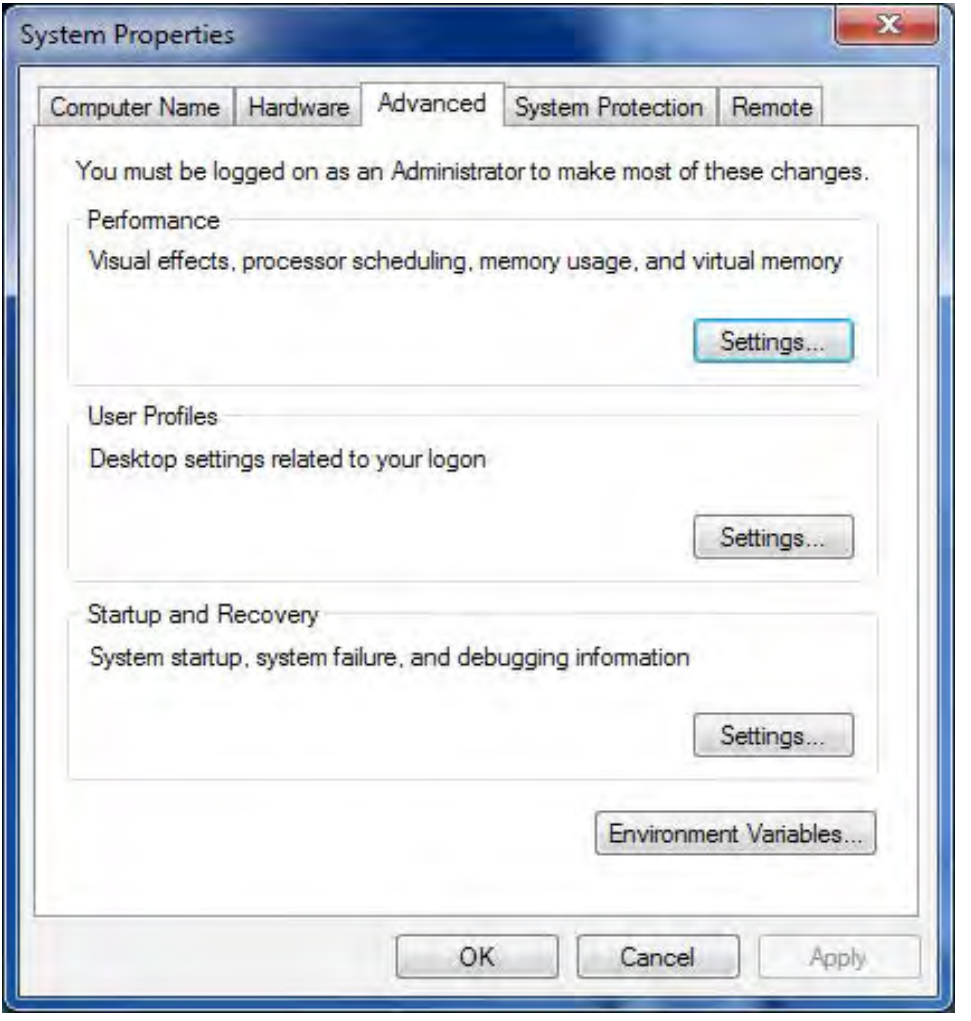


شکل ۲-۶: منوی زمینه‌ای با راست کلیک روی Computer



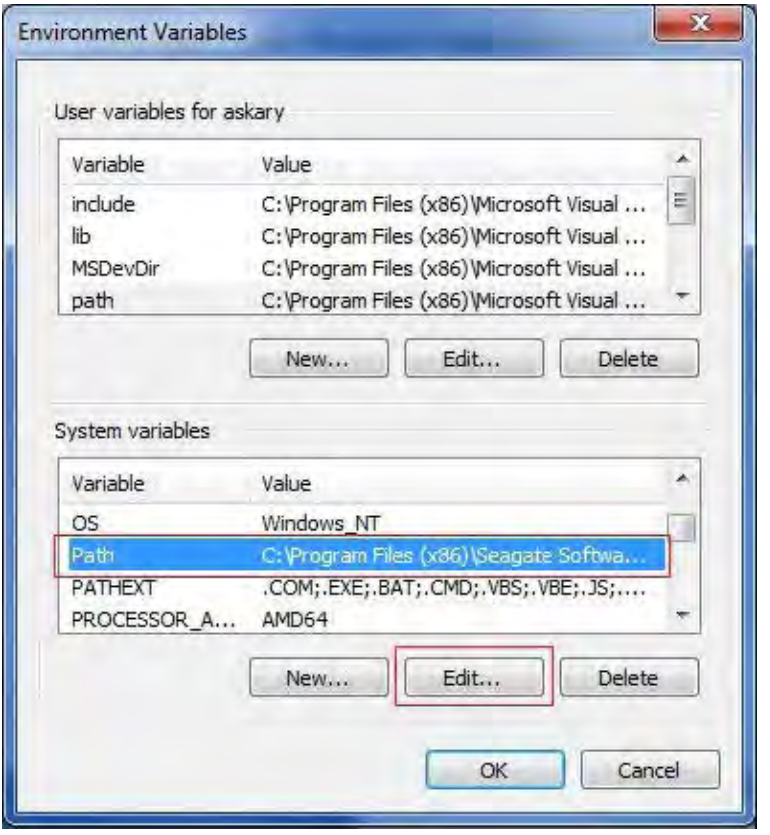
شکل ۲-۷: کادر محاوره‌ای system

روی Advanced system settings کلیـک کنیـد. کادر محـاوره‌ای system properties بـاز می‌شـود.



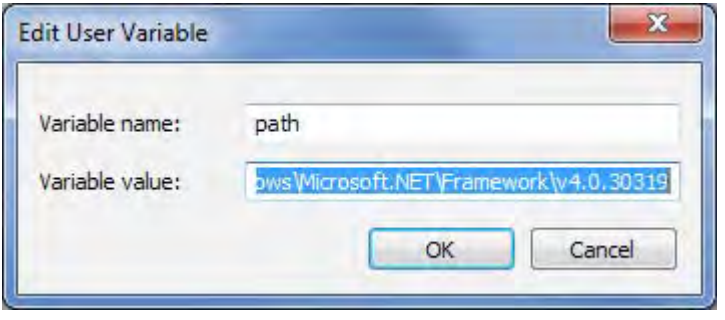
شکل ۲-۸: کادر محاوره‌ای system properties

روی دکمه‌ی Enviroment Variables کلیک کنید تا کادر محاوره‌ای آن باز شود.



شکل ۲-۹: کادر محاوره‌ای Enviroment Variable

در این کادر از لیست System variables، روی path کلیک کنید. در این قسمت تمام مسیرهای معرفی شده به سیستم‌عامل قرار دارد. مسیرها با علامت ; از یکدیگر جدا شده‌اند. برای ویرایش این مسیرها یا اضافه کردن مسیر جدید روی دکمه ...Edit کلیک کنید. کادر محاوره‌ای Edit User Variable باز می‌شود.



شکل ۲-۱۰: کادر محاوره‌ای Edit User Variable

در ایـن کادر در کادر متـن Variable Value کلیـک کـرده و بـه انتهـای مسـیرهای نوشـته شـده در آن برویـد(از کلیـد end صفحـه کلیـد اسـتفاده کنیـد.). علامـت ; قـرار داده، مسـیر فایـل csc.exe کامپیوترتـان را در ایـن قسـمت بنویسـید(مسـیری مشـابه C:\Windows\Microsoft.NET\Framework\v4.0.30319 کـه بـا توجـه به نسـخه دات نـت سیسـتم شـما، قسـمت آخـر کـه شـماره نسـخه دات نـت اسـت تغییـر می‌کنـد.).

پـس از انجـام ایـن مراحـل سیسـتم شـما مسـیر فایـل csc.exe را می‌شناسـد و هنـگام اسـتفاده از ایـن فایـل نیـاز بـه معرفـی مسـیر آن ندارید؛ بنابرایـن می‌توانیـد بـه راحتـی از آن در خـط فرمـان (دسـتور cmd) اسـتفاده کنیـد.

اگـر برنامـه ویـژوال اسـتودیو روی سیسـتم شـما نصـب اسـت، می‌توانیـد از خـط فرمـان ایـن برنامـه بـه جـای خـط فرمـان ویندوز اسـتفاده کنیـد. در صـورت اسـتفاده از ایـن محیـط نیـاز بـه انجـام مراحـل بـالا بـرای معرفـی مسـیر فایـل csc.exe نیسـت؛ زیـرا در ایـن محیـط فایـلcsc.exe شـناخته شـده اسـت. خـط فرمـان ویـژوال اسـتودیو را می‌توانیـد از منـوی start وینـدوزMicrosoft Visual Studio → All Programs → Visual Studio Tools→Developer Command prompt for … بـاز کنیـد.(بـه جـای کلمـه *Version* در هرسیسـتم شـماره نسـخه نصـب شـده قـرار دارد. در شـکل ۲-۱۱ ایـن عـدد ۲۰۱۲ اسـت و مسـیر بـه صـورت Microsoft Visual Studio 2012 \Visual Studio Tools اسـت.)

شکل ۲-۱۱: لیست محیط های خط فرمان برنامه ویژوال استودیودر منوی start ویندوز

۲-۶-۱- یادآوری دستورات خط فرمان

- دسـتور تغییـر درایـو: بـرای تغییـر درایـو، نـام درایـو را نوشـته از علامـت: اسـتفاده می‌کنیـم ماننـد :d

```
C:\windows>d:
D:\>
```

• Cd (دستور تغییر مسیر): برای کار روی فایل‌ها و یا اجرای یک فایل در خط فرمان باید نام فایل را به همراه مسیر آن در خط فرمان بنویسیم و یا ابتدا به مسیر فایل مورد نظر رفته و سپس نام فایل را بدون مسیر آن بنویسیم.

مثال: برای اجرای فایل test.exe که در مسیر d:\folder1\c# قرار دارد، می‌توان به یکی از روشهای زیر عمل کرد.

اجرای فایل با ذکر مسیرفایل d:\> d:\folder1\c#\ test.exe	تغییر مسیر خط فرمان d:\> cd d:\folder1\c# اجرای فایل d:\folder1\c#> test.exe

در ساختار درختی پوشه‌ای که در سطح بالاتر (پوشه والد) قرار دادرد، با علامت دو نقطه(..) مشخص می‌شود. به دستورات زیر توجه کنید.

رفتن به پوشه بالا	D:\folder1\c#\test> cd ..
رفتن به دو پوشه بالاتر	D:\folder1\c#> cd ..\..
D:\>	

توجه: دستور cd درایو را تغییر نمی‌دهد. در صورتی که درایو مسیر فایل شما با درایو اعلان خط فرمان متفاوت است، باید علاوه بر دستور تغییر مسیر از دستور تغییر درایو نیز استفاده کنید.

```
C:\windows> cd d:\folder1\c#
C:\windows>d:
D:\folder1\c#>
```

• dir (دستور مشاهده لیست فایل‌های یک مسیر): برای اطمینان از اینکه فایل مورد نظر در مسیر اعلان خط فرمان قرار دارد یا خیر، می‌توان از این دستور استفاده کرد که لیست تمام پوشه‌ها و فایل‌های موجود در مسیر اعلان خط فرمان را نمایش می‌دهد.

برای محدود کردن این لیست می‌توان از پارامتر در دستور dir استفاده کرد و با استفاده از علایم جایگزینی ?,* لیست را محدود کرد. برای مشاهده فایل‌هایی که پسوند cs دارند، دستور dir *.cs به کار می‌رود.

۲-۶-۲-ترجمه برنامه با مترجم csc

پس از نوشتن برنامه از منوی start → run دستور cmd را جرا کنید، تا پنجره خط فرمان باز شود. اعلان خط فرمان مسیر پوشه کاربری است که با آن کاربر به ویندوز وارد شده‌اید[۱۹]. با استفاده از دستور cd اعلان خط فرمان را به مسیری منتقل کنید که فایل اولین برنامه را در آن ذخیره کرده‌اید. ما فایل برنامه خوش آمدگویی را در مسیر d:\vs_example\firstProgram با نام welcome.cs ذخیره کرده‌ایم.

```
C:\Users\askary>d:

D:\>cd vs-example\firstProgram
```

شکل ۲-۱۲: تغییر مسیر اعلان خط فرمان

اگر مسیر فایل csc را به مسیر path سیستم اضافه نکرده باشیم، هنگام فراخوانی فایل csc خطای عدم شناخت دستور csc صادر می‌شود.

```
D:\vs-example\firstProgram>csc welcome.cs
'csc' is not recognized as an internal or external command,
operable program or batch file.
```

شکل ۲-۱۳: خطای عدم شناسایی دستور csc به دلیل مشخص نکردن مسیر فایل csc.exe

در صورت پیدا نکردن فایل برنامه برای ترجمه کردن، خطای پیدا نکردن فایل کد برنامه صادر می‌شود. دقت کنید که اگر پسوند فایل برنامه را ذکر نکنید، مترجم نمی‌تواند فایل مورد نظر را پیدا کند.

```
D:\vs-example\firstProgram>c:

C:\Users\askary>csc d:\vs-example\firstprogram\welcome
Microsoft (R) Visual C# Compiler version 4.0.30319.17929
for Microsoft (R) .NET Framework 4.5
Copyright (C) Microsoft Corporation. All rights reserved.

error CS2001: Source file 'd:\vs-example\firstprogram\welcome' could not be
       found
warning CS2008: No source files specified
```

شکل ۲-۱۴: خطای حاصل از نادرست بودن مسیر فایل برنامه

در صورتی که برنامه از نظر کامپایلر خطایی نداشته باشد، پس از ترجمه فایلی هم

۱۹ Log in

نـام بـا فایـل اصلـی و پسـوند exe (در مثـال بـالا welcome.exe)در مسـیر اعـلان خـط فرمـان سـاخته می‌شـود. در شـکل بـالا فایـل exe در مسـیر c:\users\askary سـاخته شـده اسـت، بنابرایـن بـا فراخوانـی فایـل welcome برنامـه اجـرا شـده و پیـام welcome students نمایـش داده می‌شـود.

```
C:\Users\askary>welcome
welcome students

C:\Users\askary>
```

شکل ۲-۱۵: اجرای برنامه welcome

۲-۶-۲-۱-سوییچ های csc

دسـتور csc دارای چندیـن سـوییچ اسـت کـه روش انجـام کار و تولیـد خروجـی را بـرای ایـن دسـتور تعییـن می‌کنـد. تعـدادی از ایـن سـوییچ‌ها کـه کاربـرد بیشـتری دارنـد، در ادامـه آمـده اسـت.

- **سـوییچ out/:** بـرای تعییـن محـل خروجـی می‌تـوان از سـوییچ out/ بـه صـورت زیـر اسـتفاده کـرد.

نـام ومسـیر فایـل برنامـه نـام و مسـیر فایـل خروجـی:csc /out
csc /out:d:\vs-example\firstprogram\welcome.exe d:\vs-example\firstprogram\welcome.cs

بـا ایـن دسـتور می‌تـوان نـام فایـل خروجـی را تعییـن کـرد، کـه می‌توانـد بـا نـام فایـل اصلـی یکـی نباشـد. بـرای سـادگی دسـتور بهتـر اسـت اعـلان خـط فرمـان را بـه مسـیر فایـل اصلـی تغییـر دهیـد کـه دسـتور سـاده شـود. در شـکل زیـر نـام فایـل خروجـی first.exe قـرار داده شـده اسـت.

```
C:\Users\askary>d:

D:\>cd vs-example\firstprogram

D:\vs-example\firstProgram>csc /out:first.exe welcome.cs
Microsoft (R) Visual C# Compiler version 4.0.30319.17929
for Microsoft (R) .NET Framework 4.5
Copyright (C) Microsoft Corporation. All rights reserved.


D:\vs-example\firstProgram>
```

شکل ۲-۱۶: استفاده از سوییچ :out/ در دستور csc

- **سوییچ target**: با استفاده از سوییچ target تعیین نوع فایل خروجی امکان‌پذیر است.

چند نوع فایل خروجی داریم که یکی از آنها Library است که فایل خروجی از نوع dll ایجاد می‌کند.

csc /target:نوع‌فایل‌خروجی نام‌ومسیرفایل‌برنامه
csc /target:library نام‌ومسیرفایل
→ خروجی این دستور فایلی با نام پروژه و پسوند dll است
csc /target:library d:\vs-example\firstprogram\welcome.cs

```
d:\vs-example\firstProgram> csc /target:library welcome.cs
Microsoft (R) Visual C# Compiler version 4.0.30319.17929
for Microsoft (R) .NET Framework 4.5
Copyright (C) Microsoft Corporation. All rights reserved.


d:\vs-example\firstProgram>
```

شکل ۲-۱۷: سوییچ target با نوع library

نتیجه‌ی اجرای دستور شکل ۲-۱۷ ایجاد فایل Welcome.dll است.

- **سـوییچ main/:** تعییـن نقطـه ورود بـه پـروژه در صـورت وجـود متـد Main در چنـد کلاس بـا اسـتفاده از سـوییچ :main/ انجـام می‌شـود.

در صورتـی کـه بیـش از یـک کلاس دارای متـد Main باشـند، بایـد نقطـه شـروع پـروژه در زمـان ترجمـه بـا اسـتفاده از سـوییچ main/ مشـخص شـود.

در حالت کلی داریم: csc نام‌کلاس.فضای‌نام:main/ نام ومسیرفایل‌برنامه
در صورتی که پروژه دارای فضای نام نباشد به صورت:

csc نام کلاس:main/ نام ومسیر فایل برنامه
csc d:\vs-example\program.cs /main:T1.program

در مثالـی کـه در مبحـث متـد main در ایـن فصـل ذکـر شـد، در فضـای نـام T1 دو کلاس program و class1 تعریـف شـده اسـت کـه هـر دو دارای متد Main هسـتند و در مثـال بـالا متـد Main کلاس program از فضـای نـام T1 بـه عنـوان نقطـه شـروع مشـخص شـده اسـت.

سوییچ main/ به اختصار به صورت m/ هم به کار می‌رود.

```
csc d:\vs-example\program.cs /m:T1.program
```

می‌تـوان از ترکیـب سـوییچ هـا و علایـم جایگزینـی هـم اسـتفاده کـرد. بـه مثـال زیـر توجـه کنیـد.

```
csc /target:library /out:Something.xyz *.cs
```

در ایـن دسـتور تمـام فایل‌هـای کـد برنامـه (فایل‌هایـی کـه دارای پسـوند cs هسـتند) را بـا هـم ترجمـه کـرده و از همـه آنهـا یـک فایـل خروجـی از نـوع فایـل کتابخانـه‌ای و بـه نـام Something.xyz می‌سـازد و در همـان مسـیری ذخیـره می‌کنـد کـه فایل‌هـای کـد قـرار دارنـد .

۲-۶-۳- خطاهای مربوط به کد برنامه

اگـر هنـگام نوشـتن برنامه‌هـا بـه زبـان C# توجـه لازم بـه قوانیـن نحـوی و قالـب برنامـه نداشـته باشـید، ممکـن اسـت بـا خطاهـای زیـر مواجـه شـوید.

- برنامه فاقد کلاس باشد.

```
static void Main()
{
      System.Console.WriteLine("Welcome Students.");
}
```

برنامه فوق را در فایلی به نام welcomeClass.cs و در مسیر d:\vs-example\ error ذخیره کرده و سپس آن را ترجمه کرده‌ایم. در نتیجه ترجمه خطایی رخ داده است(شکل۱۸-۲). قالب بیان خطای رخ داده در فایل csc به صورت زیر است.

توضیحات :شماره خطا :(شماره ستون , شماره خط) :نام فایل
Error\welcomeClass.cs(1,21): error cs1518: Expected class, delegate, enum, interface, or struct

شماره خط و ستون خطا در پرانتز (1,21) ذکر شده است. عدد اول شماره خط (خط ۱) و عدد دوم شماره ستون (ستون ۲۱) است و محل خطا را در کد تعیین می‌کند، سپس بعد از علامت : شماره خطا و دوباره علامت : و سپس توضیح خطا آمده است.

```
D:\vs-example>csc /out:error\welcomeclass  error\welcomeclass.cs
Microsoft (R) Visual C# Compiler version 4.0.30319.17929
for Microsoft (R) .NET Framework 4.5
Copyright (C) Microsoft Corporation. All rights reserved.

error\welcomeClass.cs(1,21): error CS1518: Expected class, delegate, enum,
        interface, or struct

D:\vs-example>
```

شکل ۲-۱۸: خطای نداشتن کلاس در برنامه

- نداشتن علامت نقطه کاما (;) در انتهای دستورات

```
class firstProgram
{
  static void Main()
  {
        System.Console.WriteLine("Welcome Students.")
  }
}
```

در ایـن برنامـه علامـت ; از انتهـای دسـتور WriteLine حـذف شـده اسـت. برنامـه را بـا نـام welcomeError.cs در همـان مسـیر d:\vs-example\error ذخیـره کـرده و ترجمـه کرده‌ایـم. در خـط ۵ و سـتون ۴۷ خطـا رخ داده است(شـکل ۲-۱۹).

(error cs1002: ; expected)

```
D:\vs-example>csc /out:error\welcomeerror  error\welcomeerror.cs
Microsoft (R) Visual C# Compiler version 4.0.30319.17929
for Microsoft (R) .NET Framework 4.5
Copyright (C) Microsoft Corporation. All rights reserved.

error\welcomeerror.cs(5,47): error CS1002: ; expected

D:\vs-example>
```

شکل ۲-۱۹: خطای نبودن علامت نقطه کاما در انتهای دستور

- خطای مشخص نکردن فضای نام بدون استفاده از using

```
class firstProgram
{
   static void Main()
   {
        Console.WriteLine("Welcome Students.");
   }
}
```

فضـای نـام System از ابتـدای دسـتور WriteLine حـذف شـده اسـت. برنامـه را بـا نـام welcomeNamespace.cs در همـان مسـیر قبلـی ذخیـره کـرده و ترجمـه کرده‌ایـم. خطـا نشـان می‌دهـد کـه کلاس Console تعریـف نشـده اسـت در خـط ۵ و سـتون ۱ خطـا رخ داده اسـت (شـکل ۲-۲۰).

(error cs0103: the name ‘Console’ does not exist in the current context)

```
D:\vs-example>csc /out:error\welcomenamespace  error\welcomenamespace.cs
Microsoft (R) Visual C# Compiler version 4.0.30319.17929
for Microsoft (R) .NET Framework 4.5
Copyright (C) Microsoft Corporation. All rights reserved.

error\welcomenamespace.cs(5,1): error CS0103: The name 'Console' does not exist
       in the current context

D:\vs-example>
```

شکل ۲-۲۰: خطای عدم شناسایی کلاس به دلیل تعیین نکردن فضای نام

با قرار دادن عبارت using System; در ابتدای برنامه، برنامه اصلاح می‌شود.

```
using System;
class firstProgram
{
    static void Main()
    {
        Console.WriteLine("Welcome Students.");
    }
}
```

۲-۷- اجرای فایل خروجی ساخته شده با مترجم

دو روش برای اجرای فایل خروجی وجود دارد.

۱- **اجرای فایل خروجی در خط فرمان:** نام فایل خروجی را به همراه مسیر آن، در خط فرمان نوشته و کلید Enter را فشار می‌دهیم. در بالا از این روش استفاده کرده‌ایم.

۲- **دابل کلیک روی نام فایل خروجی:** در این حالت پنجره کنسول باز شده، پیام نمایش داده شده و بلافاصله پنجره بسته می‌شود؛ بنابراین نمی‌توان خروجی را مشاهده کرد. برای رفع این مشکل باید از دستور ReadKey استفاده کنیم که از متدهای کلاس Console است. این دستور برای دریافت کلیدی از صفحه کلید منتظر می‌ماند. به این شکل برنامه تا فشرده شدن کلیدی دلخواه متوقف می‌شود و کاربر می‌تواند پیام‌ها و مطالب نوشته شده در پنجره کنسول را مشاهده کند. با فشار یکی از کلیدهای صفحه کلید پنجره بسته می‌شود. قبل

از ایـن دسـتور بهتـر اسـت پیـام"press a key to exit...." را بـرای اطـلاع کاربـر روی صفحـه نمایـش نشـان دهیـد.

برنامـه welcome.cs را بـرای اجـرا بـا دابـل کلیـک روی نـام فایـل خروجـی بـه شـکل زیـر تغییـر داده‌ایـم.

```
using System;
class firstProgram
{
    static void Main()
    {
        Console.WriteLine("Welcome Students.");
        Console.WriteLine("press a key. To exit…..");
        Console.ReadKey();
    }
}
```

۲-۸- فضاهای نام .Net

کتابخانـه دات نـت دارای چندیـن فضـای نـام بـرای سـازماندهی کلاس‌هـای آمـاده اسـت. چنـد نمونـه از ایـن فضاهـای نـام عبارتنـد از:.

جدول ۱-۲: چند فضای نام در کتابخانه دات نت

Microsoft.Win32	ایـن فضـای نـام امـکان مدیریـت رویدادهـای سیسـتم‌عامل از جملـه دسـتکاری رجیسـتری سیسـتم، نمایـش فایـل و مدیریـت سیسـتم عامـل را فراهـم می‌کنـد.
Microsoft.Windows	ایـن فضـای نـام از انـواع تم‌هـا و پیش‌نمایـش برنامه‌هـای کاربـردی از نـوع WPF پشـتیبانی می‌کنـد.
System.Net	ایـن فضـای نـام بـرای بیشـتر پروتکل‌هایـی کـه امـروزه در شـبکه اسـتفاده می‌شـوند، یـک رابـط برنامه‌نویسـی سـاده فراهـم می‌کنـد.
System	فضـای نـام سیسـتم شـامل کلاس‌هـای اساسـی و کلاس‌هـای پایـه بـرای تعریـف انـواع داده‌هـا، رویدادهـا و مدیریـت رویدادهـا، رابط‌هـا و..... اسـت.

کلاس Console کـه در ایـن فصـل از آن اسـتفاده کردیـم، یکـی از کلاس‌هـای فضـای نـام System اسـت. برخـی از کلاس‌هـای ایـن فضـای نـام در ادامـه آمـده اسـت.

جدول ۲-۲: برخی از کلاس‌های فضای نام System

Array	برای ایجاد، مدیریت، مرتب‌سازی و کار روی آرایه‌ها است.
Convert	برای تبدیل انواع داده‌ها به یکدیگر استفاده می‌شود.
Environment	اطلاعاتـی دربـاره محیـط فعلـی، پلـت فـرم آن و امـکان تغییـر تنظیمـات آن را فراهـم می‌کنـد.
Math	متدهایـی بـرای انجـام توابـع مثلثاتـی و لگاریتمـی و توابـع ریاضـی دیگـر فراهـم می‌کنـد.
Object	کلاسـی بـرای یکـی از انـواع داده.(در فصـل ۴ انـواع داده آمـده اسـت.)
String	ایـن کلاس امـکان نمایـش متن‌هایـی از کارکترهـای یونیکـد و کار روی آن‌هـا را فراهـم می‌کنـد.(یکـی از انـواع داده‌هـا در C# اسـت کـه در فصـل ۴ بـه آن می‌پردازیـم.)
StringCompare	ایـن کلاس متدهایـی بـرای مقایسـه بیـن رشـته‌ها فراهـم می کنـد .
TimeZone	برای نمایش time zone استفاده می‌شود.
ValueType	کلاس پایه برای انواع داده‌ها است.

۲-۸-۱- کلاس Console

کلاس Console بـرای نمایـش ورودی و خروجی‌هـای اسـتاندارد و خطاهـای رخ داده در برنامه‌هـای Console Application اسـت. همان‌طورکـه در اولیـن برنامـه گفتـه شـد، کلاس شـامل مجموعـه‌ای از داده‌هـا و عملیـات اسـت کـه بـه ترتیـب بـه آنهـا ویژگی‌هـا و متدهـای کلاس گفتـه می‌شـود. کلاس Console دارای ویژگـی و متدهـای زیـادی اسـت کـه در اینجـا بـه بیـان برخـی از آنهـا می‌پردازیـم.

۲-۸-۱-۱-ویژگیهـای کلاس Console بـرای تعییـن رنـگ زمینه و قلـم و پـاک کـردن صفحـه

BackgroundColor: رنگ زمینه کنسول را مشخص می‌کند.

ForegroundColor: رنگ قلم را در کنسول مشخص می‌کند.

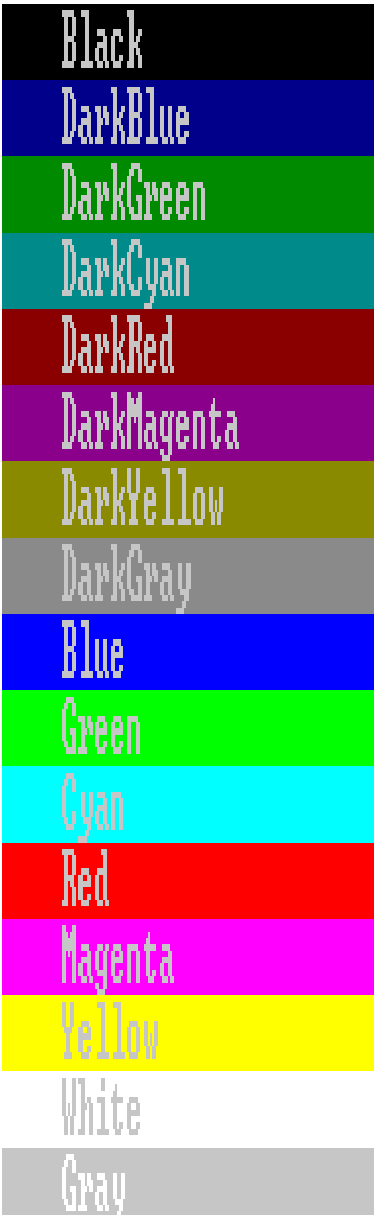
برای استفاده از این مشخصات باید قبل از مشخصه نام کلاس را ذکر کرد.

برای تغییر رنگ زمینه: Console.BackgroundColor = رنگ دلخواه

برای تغییر رنگ قلم: Console.ForegroundColor = رنگ دلخواه

C# دارای یـک جعبـه رنـگ ۱۶ تایـی بـا نـام ConsoleColor در فضـای نـام System اسـت؛ بنابرایـن بـه صـورت System.ConsoleColor اسـتفاده می‌شـود و در صـورت اسـتفاده از دسـتور using System; می‌تـوان بـدون نوشـتن فضـای نـام System از آن اسـتفاده کـرد. از ایـن جعبـه رنـگ بـرای تغییـر رنـگ زمینـه یـا قلـم کنسـول می‌تـوان اسـتفاده کـرد. نـام ایـن ۱۶ رنـگ در جـدول زیـر آمـده اسـت.

جدول ۳-۲: رنگ‌های ConsoleColor

نام رنگ در ConsoleColor	نام رنگ	نمونه رنگ
Black	مشکی	
DarkBlue	سورمه‌ای	
DarkGreen	سبز تیره	
DarkCyan	فیروزه‌ای تیره	
DarkRed	قرمزتیره	
DarkMagenta	بنفش	
DarkYellow	زرد تیره	
DarkGray	خاکستری تیره	
Blue	آبی	
Green	سبز	
Cyan	فیروزه‌ای	
Red	قرمز	
Magenta	صورتی	
Yellow	زرد	
White	سفید	
Gray	خاکستری	

برای استفاده از هر رنگ باید قبل از نام رنگ، نام جعبه رنگ آورده شود. برای مثال ConsoleColor.Blue رنگ آبی را نشان می‌دهد.

دو دستور زیر رنگ زمینه را آبی و رنگ قلم را سفید قرار می‌دهد.

```
Console.ForegroundColor = ConsoleColor.White;
Console.BackgroundColor = ConsoleColor.Blue;
```

دو دستور زیر رنگ فعلی قلم و زمینه را ذخیره می‌کند.

```
ConsoleColor currentBackground = Console.BackgroundColor;
ConsoleColor currentForeground = Console.ForegroundColor;
```

البته در دو دستور بالا متغیرهای currentBackground و currentForeground متغیرهای تعریف شده به وسیله‌ی برنامه‌نویس از نوع ConsoleColor است. در فصل۴ کتاب در مورد انواع متغیرها و تعریف آنها بحث می‌شود.

۲-۸-۱-۲- متدهای کلاس Console

کلاس کنسول دارای عملیات‌ی روی کنسول می‌باشد که می‌توان از آنها در کد برنامه به صورت "نام متد.Console" استفاده کرد. برخی از آنها در ادامه آمده است.

- **Clear()**: برای پاک کردن زمینه کنسول استفاده می‌شود که زمینه را با رنگی که در ویژگی BackgroundColor آمده است، پاک می‌کند. برای استفاده از آن دستور زیر را به کار می‌بریم.

```
Console.Clear();
```

توجه: هنگام استفاده از متدها دقت کنید که متدها دارای ( ) هستند، حتی اگر مانند متد clear آرگومان ورودی نداشته باشند.

به تکه کد زیر توجه کنید.

```
Console.BackgroundColor = ConsoleColor.Blue;
Console.Clear();
```

بعد از اجرای این دو دستور زمینه کنسول به رنگ آبی می‌شود.

- **ResetColor()**: اگر رنگ زمینه و قلم کنسول را تغییر داده باشید با این متد می‌توانید آنها را به رنگ پیش‌فرض برگردانید.
- **ReadKey()**: استفاده از این متد سبب می‌شود که برنامه تا فشرده

شدن یک کلید از صفحه کلید متوقف شود.

- **Beep(Int32, Int32)**: برای پخش صدای بیب در فرکانس مشخص برحسب هرتز و در مدت زمان مشخص برحسب میلی ثانیه از طریق بلندگوی کنسول از این متد استفاده می‌شود. ورودی اول فرکانس برحسب هرتز و ورودی دوم زمان برحسب میلی ثانیه است.

Console.Beep(فرکانس, زمان)

برای پخش صدای بیپ در مدت یک ثانیه از دستور Console.Beep(2000,1000) استفاده می‌کنیم.

- **WriteLine(String)**: این متد برای نوشتن پیام روی صفحه نمایش استفاده می‌شود. این دستور به انتهای پیام علامت پایان خط جاری را اضافه می‌کند؛ بنابراین پس از نوشتن پیام، مکان‌نما به خط بعد منتقل می‌شود.

- **WriteLine ()**: این متد مکان‌نما را به خط بعد منتقل می‌کند. اگر خط جاری خالی باشد، یک خط خالی دیده می‌شود ولی اگر مکان‌نما در مکانی به غیر از ابتدای خط جاری باشد بقیه‌ی خط خالی می‌ماند.

- **Write(String)**: این متد مشابه متد WriteLine است با این تفاوت که به انتهای پیام چیزی اضافه نمی‌شود؛ بنابراین مکان‌نما در انتهای پیام نوشته شده روی صفحه نمایش باقی می‌ماند و پیام بعدی در ادامه آن نوشته می‌شود، درحالی که پیام‌های نوشته شده با دستور WriteLine هر کدام در یک خط جداگانه قرار می‌گیرد.

در صورتی که ما چندین دستور Console.Write را پشت سر هم بنویسیم، خروجی‌ها تا پر شدن خط جاری در همان خط قرار می‌گیرد و پست سر هم نمایش داده خواهند شد. برای مثال:

```
Console.Write("A");
Console.Write("B");
Console.Write("C");
```

خروجی کد بالا به صورت روبرو خواهد بود: ABC

اگر به جای دستور Console.Write از دستورConsole.WriteLine استفاده کنیم، خواهیم داشت:

```
Console.WriteLine("A");
Console.WriteLine("B");
Console.WriteLine("C");
```

خروجی دستورات بالا به صورت زیر است:

```
A
B
C
```

برای درک بهتر مثالی با دو دستور Write و WriteLine را بررسی می‌کنیم:

```
Console.Write("A");
Console.WriteLine("B");
Console.WriteLine("C");
Console.Write("D");
Console.Write("E");
```

خروجی کد بالا به صورت زیر خواهد بود:

```
AB
C
DE
```

برنامه زیر مثال دیگری از دو متد Write و WriteLine است.

```
using System;
class Program
{
    static void Main()
    {
        Console.WriteLine("first line");
        Console.Write("hello students ");
        Console.Write("welcome");
        Console.WriteLine(" line 2");
        Console.WriteLine("a" + "b" + "c");
        Console.Write("d");
        Console.Write("e");
        Console.WriteLine();
        Console.ReadKey();
    }
}
```

خروجی فایل مطابق شکل ۲-۲۱ است.

دستور شماره ۱ پیام را در خط اول و دستورات شماره ۲و ۳و ۴ پیام را در خط دوم و دستور ۵ پیام را در خط سوم ودستورات ۶ و ۷ پیام را در خط چهارم می‌نویسند.

```
first line
hello students  welcome line 2
abc
de
```

شکل ۲-۲۱: خروجی مثال دستور Write وWriteLine

روش کار با دستورات Write و WriteLine و دستورات ورودی چون ReadLine در فصل چهار به صورت مفصل تر گفته شده است

واژگان

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	Comment	توضیح
۲	Common Intermediate Language	زبان واسط مشترک
۳	Common Type System	سیستم نوع مشترک
۴	Just in Time Compiler	مفسر در لحظه
۵	Key word	کلمه کلیدی
۶	Managed code	کد مدیریت شده
۷	Microsoft Intermediate Language	زبان واسط مایکروسافت
۸	Reserved word	کلمه رزرو شده
۹	Source code	کد اصلی
۱۰	User Defined	تعریف شده به وسیله‌ی کاربر

پرسش و پاسخ

۱ آیا باید حتماً Net Framework. نصب باشد تا کد در خط فرمان جواب بدهد؟

بله، Net Framework. بصورت خودکار در هنگام نصب ویندوز هفت و بالاتر، نصب شده است ولی برای نسخه‌های ویندوز XP و پایین‌تر باید نصب گردد. همچنین باید مسیر آن در متغیر Path تعریف شود، تا در خط فرمان قابل اجرا باشد.

۲ آیا نصب کردن notepad++ اجباری است؟

خیر، داشتن ویرایشگر ساده همانند Notpad نیز کفایت می‌کند ولی++Notpad می‌تواند برای مدیریت بلاک‌ها، نحوه نگارش، قرارگیری کلمات و عبارات به صورت درست کمک کند.

۳ آیا لازم است مفهوم Framework برای دانش‌آموزان بیان شود؟

بله - Framework در واقع مجموعه‌ی منسجم از کتابخانه‌ی کلاس‌ها و توابع از پیش تعریف شده است که قابلیت‌های بالقوه‌ی گوناگون از یک زبان برنامه‌نویسی را در خود دارد و بدین ترتیب کاربر نهایی را قادر می‌سازد که از امکانات یک زبان استفاده کند بدون اینکه درگیر مسائل پیچیده و وقت‌گیر آن شود؛ لذا همان طور که یک کلاس یا تابع با هدف جلوگیری از تکرار مکررات و افزایش سرعت کار تعریف می‌شود، Framework از این هم فراتر رفته و علاوه بر افزایش سرعت، مواردی مثل توسعه‌پذیری و ساده‌سازی را هم مد نظر دارد.

۴ مفهوم کلاس و شی چگونه بیان شود؟ از چه مثالی برای توضیح آنها استفاده شود؟

با یک مثال مفهوم کلاس به شرح زیر است.

در یک شیرینی‌پزی برای ساخت شیرینی‌ها نیاز به قالب هست، قالب شیرینی دقیقا مشابه شیرینی است ولی خود شیرینی نیست ولی برای ساخت شیرینی به آن نیاز است. کلاس هم مشابه این قالب‌ها در برنامه استفاده می‌شود به عنوان مثال دانش‌آموز دارای یک سری مشخصات و رفتار است. برای ساده کردن کار می‌توانیم در برنامه‌نویسی یک کلاس، دانش‌آموز ایجاد کرده مثل یک قالب و از طریق آن نمونه‌های دانش‌آموزان را ایجاد کنیم.

۵ آیا یک برنامه بدون namespace قابل اجرا است؟

بله، فضای نامی در برنامه همانند یک پوشه عمل می‌کند.(کمک به دسته‌بندی قسمت‌های مختلف برنامه)

۶ آیا با اجرای دستور BackgroundColor حتما باید دستور Clear نیز استفاده شود تا رنگ پشت زمینه نوشته‌ها تغییر کند؟

خیر، با تغییر مشخصه BackgroundColor رنگ زمینه نوشته‌هایی تغییر می‌کنند که بعد از اجرای این دستور در صفحه، نمایش داده خواهند شد.

با اجرای دستور Clear تمامی فضای صفحه نمایش با کد فضای خالی پر خواهد شد.

۷ در تمرین ۴ و ۵ با توجه به تعریف عملگرها که در این فصل توضیح داده نشده است، بهتر نبود این دو تمرین در فصل عملگرها گفته می‌شد؟

هدف مسئله، عملگر و محاسبات نیست، بلکه فقط استفاده از دستور WriteLine است، حتی شکل دستور WriteLine در سوال ۴ نیز آمده است.

۸ چگونه مسیر خروجی کامپایلر را تنظیم نماییم؟

سوییچ out مکان خروجی را به مسیر متفاوت از مسیر جاری تغییر می‌دهد

واحد کار

واحد کار شماره: ۲-۱	فصل ۲ - صفحه ۲۳ تا ۳۶	مدت آموزش: ۹۰ دقیقه	روش تدریس پیشنهادی: پرسش و پاسخ
عنوان: نمایش جمع دو عدد		هدف کلی: تمرین متد های Write،WriteLline و ویژگی های ForegroundColor و BackgroundColor	
اهداف رفتاری: انتظار می رود دانش آموز ۱. بتواند متن را به صورت صحیح در خروجی نمایش دهد. ۲. متن و غیر متن را در جای صحیح و به نحو درست بکار ببرد. ۳. محیط کنسول را تغییر رنگ بدهد		اهداف پنهان درس:درک ضمنی تفاوت ویژگی و متد- .درک احساس نیاز به وجود ورودی های مختلف و پذیرش آنها توسط یک متد هدف ایده آل: تولید برنامه های خلاقانه به کمک جعبه رنگ	

فرایند تدریس:

از یک دانش آموز می خواهیم با دو ماژیک غیر همرنگ، متنی دو خطی را روی تخته بنویسد بطوریکه هر خط دارای متنی با دو رنگ متفاوت باشد.

در ضمن از وی می خواهیم که پایان خط را مشخص نماید. با پرسش سوالات زیر ذهن دانش آموزان را متوجه نکات مهم می کنیم:

- کجا خط جاری باید منتقل خط بعدی شود؟ آیا با تغییر رنگ، خط جاری تغییر پیدا کرده است؟ این متن، به سه قسمت تقسیم می شود. این سه بخش کدامند؟

پس از آن برنامه ای می نویسیم که نام دانش آموز پای تخته را در خروجی نمایش دهد. این سوال را می پرسیم که روی تخته برای تغییر رنگ چه کرده ایم؟پس از گرفتن جواب، با انجام تغییرات روی کد، به آنها نمایش می دهیم چگونه می توانیم در C# همین کار را انجام بدهیم.

از دانش آموزان می خواهیم مراحل زیر را انجام دهند

۱. نام خود را در یک خط نمایش دهند ۲. نام خود را آنطور که می خواهند رنگ بزنند! ۳.در خط بعد جمع دو عدد را به صورت ?=?+? نمایش دهند ۴. نتیجه را با رنگ دیگری مشخص نمایند.

پس از نوشتن برنامه، توجه دانش آموزان را به جمع هر دو عدد دلخواه جذب می کنیم. بدین ترتیب آمادگی برای یادگیری دریافت ورودی را ایجاد خواهیم کرد.

واحد کار شماره: ۲-۱	فصل ۲ - صفحه ۳۳ تا ۳۶

توصیه های اجرایی:

- مراحل کار را با هم درنیامیزید. می توان به صورت کلی هدف را بگوییم. اما به صورت مرحله به مرحله کار را از آنها بخواهیم. به این صورت کار بهتر انجام شده و دانش آموز هم با مفهوم توسعه بیشتر آشنا می شود.
- معمولا دانش آموزان دوست دارند نام خود را برجسته تر و مشخص تر نمایش بدهند .آزاد گذاشتن آنها برای تغییر رنگ نام و یا تغییرات دیگر می تواند در جهت یادگیری درس موثر باشد.
- در جلسات اول یادگیری، آزمون و خطای زیاد مشاهده می شود.. در عین آزاد گذاشتن دانش آموزان می شود آنها را توصیه به استفاده از مرجع نمود

ارزشیابی از انتظارات: (سوالات به صورت ص غ هستند)

۱. با استفاده از WriteLine نمایش متن بعدی، در خط بعد خواهد بود.

۲. برای نمایش متن ها در خط جاری WriteLine مناسب می باشد.

۳. برای تغییر رنگ قلم از ویژگی ForegroundColor استفاده می شود.

۴. مقدار یک رنگ را می توانیم از کلاس ConsoleColor بدست بیاوریم.

۵. رنگ زمینه کنسول قابل تغییر نیست.

۶. Write() یک متد است.

۷. متن را در داخل دابل کوتیشن (« ») می نویسیم.

۸. متد ReadKey() باعث می شود برنامه بسته شود.

تکالیف و پروژه های پیشنهادی:

۱. پرچم ایران را در محیط کنسول شبیه سازی کنید.

۲. ضرب دو عدد را به صورت مقابل نمایش دهید و هر عدد را با یک رنگ خاص مشخص نمایید.نتیجه کار باید دارای پس زمینه رنگی باشد .

نمونه سوال از فصل ۲

درستی یا نادرستی هر عبارت را تعیین کنید. **د** برای درست و **ن** برای نادرست.

۱ - در نام گذاری یک کلاس نباید بین کلمات فاصله گذاشته شود.

۲ - استفاده از متد Main در هر برنامه به زبان C# اختیاری است.

۳ - برای درج توضیحات بیش از یک خط از علامت // استفاده می‌شود.

۴ - برای تغییر رنگ زمینه کنسول باید ForegroundColor را مقداردهی کرد.

۵ - نام متد main در برنامه حتما باید با M نوشته شود.

در سوالات زیر گزینه صحیح را انتخاب کنید.

۶- کدامیک از گزینه های زیر از ویژگیهای زبان C# است؟

الف) شیءگرا و سطح بالا　　ب) شیءگرا و سطح پایین
ج) سطح پایین و همه منظوره　　د) سطح میانی و همه منظوره

۷- در عبارت;System.Console.WriteLine("Hello") کدام فضای نام است؟

الف) System　　ب) Console　　ج) WriteLine　　د) Hello

۸-برای نمایش پیام بر روی صفحه نمایش، متن را باید درون چه علامتی قرار دهیم؟

الف) ‘ ‘　　ب) " " (نقل قول)　　ج) , , (کاما)　　د) ! (علامت تعجب)

۹-کدامیک از گزینه های زیر یک کلاس آماده در زبان C# است؟

الف) Console　　ب) Main　　ج) void　　د) static

۱۰- چنانچه پس از ترجمه خطایی مشاهده نشود و عمل ترجمه موفقیت آمیز باشد، در مسیر برنامه فایل با کدام پسوند ایجاد می شود؟

الف) .bat　　ب) .exe　　ج) .txt　　د) .net

در جملات زیر، جاهای خالی را با عبارات مناسب پرکنید.

۱۱-برای درج توضیحات یک خطی در برنامه از علامت ..... استفاده می‌شود.

۱۲-برای تعریف یک کلاس جدید در زبان برنامه نویسی C# از کلمه کلیدی..... استفاده می شود.

۱۳-پسوند فایل‌های برنامه به زبان Csharp ..... است

۱۴-متدی که منتظر زدن یک کلید از کاربر است..... نام دارد.

۱۵- برای پاک کردن صفحه از متد ..... استفاده می‌شود

سوال جور کردنی

برای هر پرسش ستون راست یک پاسخ از ستون چپ پیدا کنید و در جای خالی بنویسید.

(.....) ۱۶- تغییر رنگ قلم — الف- using

(.....) ۱۷-معرفی یک فضای نامی — ب- ConsoleColor

(.....) ۱۸-تغییررنگ زمینه — ج- ویژگی ForegroundColor

(.....) ۱۹- چاپ یک خط خالی — د- ویژگی BackgroundColor

ه- WriteLine()

Trace و خطایابی

۲۰- هریک از دستورات زیر چه عملی انجام می دهند؟

```
System.Console.WriteLine("Hello");

System.Console.WriteLine();
```

۲۱- برنامه زیر چه اشکالی دارد؟

```
static void main()
{
  System.Console.WriteLine("Hello");
}
```

ارتباط تمارین فصل و اهداف رفتاری

ردیف	اهداف رفتاری فصل دوم	خودآزمایی	تمرینات برنامه‌نویسی
۱	قالب کلی یک برنامه ساده در زبان C# را بیان کند.	۱و۲و۳	
۲	نحوه تعریف یک کلاس و متد را در یک برنامه ساده به کار بندد.	۱۰	۱
۳	با استفاده از یک ویرایشگر،برنامه ی ساده بنویسد و آن را ذخیره کند .		۱ تا ۷
۴	برنامه ذخیره شده را با استفاده از مترجم در پنجره کنسول ترجمه کرده، سپس اجرا نماید.		۱ تا ۷
۵	با استفاده از متدهای مربوط به رنگ، تغییری در خروجی برنامه ی خود به وجود آورد.	۱۱	۷

بودجه بندی فصل ها

فصل دوم
هفته چهارم دی ماه و اول بهمن ماه

موضوع : آشنایی با زبان C#

هدف کلی: آشنایی با قالب کلی یک برنامه کنسول ساده و نحوه تعریف کلاس و متد - استفاده از چند متد کاربردی

ساعات تئوری: ۴ **ساعات عملی:** ۸

فعالیتهای آموزشی موازی
-بررسی و حل تمرینهای فصل اول
-مرور واژگان
- اجرای برخی برنامه های نوشته شده توسط C#

فعالیتهای رقابتی و ارزشیابی
-بررسی و ارزیابی تحقیقات دانش آموزان
-ارزشیابی از فصل اول، حل سوالات چندگزینه ای به شیوه ی گروهی و رقابتی

فصل ۳

آشنایی با ویژوال استودیو

اهداف رفتاری

IDE را تعریف کند و برای آن مثال بیاورد.

مزایای استفاده از IDE و کاربرد VS را بیان نماید.

یک برنامه جدید در محیط VS از نوع کنسول ایجاد کند، آن را ذخیره، ترجمه و اجرا نماید.

آشنایی با ویژوال استودیو

کلید واژه‌ها: IDE، ویژوال استودیو، نصب ویژوال استودیو، ترجمه و اجرای برنامه در VS

مقدمه

در فصل دوم با الگوی اولین برنامه C# آشنا شدید و آن را در ویراستیاری مانند Notepad++ نوشته، در خط فرمان کامپایل کردیم و در صورت وجود خطا آن را برطرف کرده، اجرا نمودیم. این روش با سختی‌هایی روبرو بود که با آن آشنا شدید. حال در این فصل می‌خواهیم به جای تمام این مراحل از یک محیط IDE[1] استفاده کنیم.

۳-۱- آشنایی با IDE

محیط یکپارچه توسعه (IDE) نرم‌افزاری است که با نام محیط تعاملی توسعه نیز شناخته می‌شود و امکانات جامعی را در اختیار برنامه‌نویسان قرار می‌دهد. محیط IDE، محیطی برای برنامه‌نویسی است که شامل ویراستیار برای نوشتن کد برنامه، مترجم، عیب‌یاب و رابط گرافیکی کاربر است؛ بنابراین تمام مراحل انجام شده در فصل ۲ را می‌توان به راحتی در آن انجام داد. در این فصل از نرم‌افزار ویژوال استودیو نسخه 2012 Express for Windows Desktop به عنوان محیط IDE استفاده می‌کنیم. این نرم‌افزار بر پایه چارچوب net. است که در فصل قبل توضیح داده شد. بنابراین با برنامه‌نویسی در این نرم‌افزار می‌توان از کلاس‌های موجود در کتابخانه و امکانات net. استفاده کرد. نسخه Express این نرم‌افزار رایگان می‌باشد و آن را می‌توان از وب سایت Microsoft و با آدرس زیر دانلود کرد و نصب نمود.

http://www.microsoft.com/en-us/download/details.aspx?id=34673

۳-۱-۱- مقایسه IDE ها

برای زبان‌های مختلف IDEهای گوناگونی ساخته شده است. به عنوان نمونه برخی زبان‌ها و IDEهای تهیه شده برای آنها فهرست شده است:

C\C++: turbo c++، xcode و code

Python: Dialog Blocks، Aptana و Shift Edit

Pascal:Lazarus و Bricx command center

[1] Integrated Development Environment

Java: NetBeans و Eclipse

PHP: komodo، Espresso و PHP Storm

C#: MonoDevelop، SharpDevelop، Xamarin Studio، Visual Studio

در جدول زیر مقایسه بین ویژوال استودیو و دو IDE دیگر را می‌بینید که برای برنامه‌نویسی #C به کار می‌روند.

جدول ۳-۱: مقایسه بین ویژوال استودیو و دو IDE دیگر

IDE	توسعه دهنده	ویندوز	لینوکس	مکینتاش	سکوهای دیگر	مجوز
Microsoft Visual Studio	Microsoft	✓	✗	✗		مالکیتی وجود نسخه رایگان
MonoDevelop	Xamarin و the Mono community	✓	✓	✓	FreeBSD OpenBSD Solaris	نرم افزار آزاد LGPL
SharpDevelop	ICSharpCode Team	✓	✗	✗		نرم افزار آزاد LGPL

۳-۲- نصب ویژوال استودیو

مراحل نصب ویژوال استودیو در ادامه به صورت تصویری آمده است.

فایل wdexpress_full فایل اجرایی برای نصب است. آن را اجرا کنید (بهتر است این کار را با گزینه Run as administrator انجام دهید.). کادر محاوره‌ای نصب باز می‌شود. (شکل ۳-۱)

شکل ۳-۱ : شروع نصب ویژوال استودیو و تعیین محل نصب

در ایـن کادر میـزان فضـای موردنیـاز بـرای نصـب ۴/۵۳ گیگابایـت ذکـر شـده اسـت و محـل نصـب را کـه بـه صـورت پیش‌فـرض c:\program files (x86)\Microsoft\Visual Studio 11.0 اسـت، می‌تـوان تغییـر داد و در پایـان گزینـه پذیـرش قوانیـن را انتخـاب کنیـد(I agree to the License terms and conditions.). در ایـن زمـان دکمـه install ظاهـر می‌شود(شـکل ۳-۲). روی آن کلیـک کنیـد.


Visual Studio
Express 2012 for Windows Desktop
Setup requires 4.53 GB in:
C:\Program Files (x86)\Microsoft Visual Studio 11.0
You must agree to the License Terms before you can install the product.
Visual Studio 2012 automatically sends information to Microsoft about your installation experience and notifies you when an update is available and connects you to content online. To learn more, see the Privacy Statement.
I agree to the License terms and conditions.
Join the Customer Experience Improvement Program to help improve the quality, reliability and performance of Visual Studio.
INSTALL

شکل ۳-۲

شکل ۳-۳

شکل ۳-۴

در برخی از سیستم‌ها در این مرحله نیاز به restart کردن سیستم است ولی در تمام سیستم‌ها restart کردن لازم نیست(شکل ۳-۴). پس از restart شدن سیستم، نصب ادامه پیدا می‌کند (شکل۳-۵).

شکل ۳-۵

شکل ۳-۶

روی LAUNCH کلیک کنید(شکل ۳-۶).

شکل ۳-۷

شکل ۳-۸

پس از تکمیل نصب برای رجیستر شدن باید شماره سریال را از سایت مایکروسافت دریافت کرده، وارد کنید و کلید next را کلیک کنید(شکل۳-۷). کادری بازشده، موفقیت‌آمیز بودن عملیات رجیستر را اطلاع می‌دهد. روی Close کلیک کنید(شکل۳-۸).

۳-۲-۱- کاربرد Removal Tool

در برخی از سیستم‌ها نصب نرم‌افزار با مشکل مواجه می‌شود و این اشکال ممکن است به علت وجود برخی از ویروس‌ها و یا سازگارنبودن نرم‌افزار با نرم‌افزارهای نصب شده روی سیستم باشد، در این صورت می‌توان با استفاده از برنامه‌های Removal Tool این مشکل را برطرف کرد. ابزار حذف نرم‌افزارهای مضر[۲] مایکروسافت کمک می‌کند تا نرم‌افزارهای مخرب را از کامپیوتر خود حذف کنید و از سیستم‌عامل‌های ویندوز۸، ویندوز سرور ۲۰۱۲، ویندوز۷، ویندوز ویستا، ویندوزسرور ۲۰۰۳، ویندوز سرور ۲۰۰۸، یا ویندوز ایکس پی پشتیبانی می‌کند. مایکروسافت هر ماه نسخه جدیدی ازاین ابزار منتشر می‌کند. پس از دانلود ابزار، ابزار یک بار اجرا می‌شود و کامپیوتر شما را از نظر وجود نرم‌افزارهای مخرب شایع خاص(از جمله بلستر، ساسر، و Mydoom) بررسی کرده، در صورت

۲ Malicious Software Removal Tool(MSRT)

یافتن نرم‌افزارهای مخرب به حذف آنها کمک می‌کند.

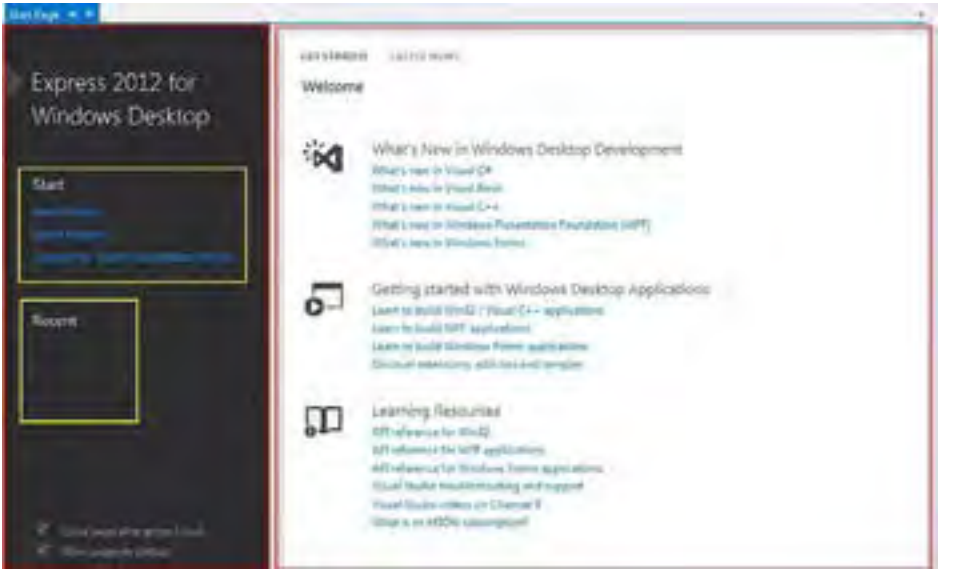
این ابزار تنها موارد خاصی را بررسی می‌کند و مشکل را برطرف می‌کند و به هیچ وجه به عنوان آنتی ویروس کاربرد ندارد. اگر امکان به‌روز شدن ویندوز در سیستم شما فعال باشد، هرماه نسخه جدید این ابزار روی سیستم شما نصب شده، سیستم را بررسی می‌کند و در صورت پیدا کردن فایل مضر به شما اطلاع می‌دهد.

۳-۳- ایجاد اولین پروژه

پس از نصب ویژوال استودیو در منوی start ویندوز مطابق شکل زیر دو منو اضافه می‌شود(شکل۳-۹). منوی Microsoft Visual Studio 2012 برای دسترسی به ابزارهای این نرم‌افزار و ابزارهای کار گروهی است و منوی Microsoft Visual Studio 2012 Express برای اجرای ویژوال استودیو به منظور برنامه‌نویسی است. برای اجرای ویژوال استودیو گزینه VS Express for Desktop را انتخاب کنید.

شکل ۳-۹ : قسمتی از منوی start ویندوز بعد از نصب visual studio 2012 Express

برنامه ویژوال استودیو اجرا می‌شود و کادر محاوره‌ای start page باز می‌شود(شکل ۳-۱۰).



شکل ۳-۱۰ : کادر start page

کادر start page دارای دو پنل است. پنل سمت راست خوش آمدگویی به کاربر است و دارای پیوندهایی برای آگاهی از تازه‌های این نرم‌افزار، چگونگی شروع کار با آن، آشنایی با راهنمای MSDN، منابع و عیب یابی در ویژوال استودیو است. پنل سمت چپ دارای قسمتهای Start و Recent است. اگر بار اولی است که ویژوال استودیو را اجرا می‌کنید، (شکل ۳-۱۰)بخش Recent خالی است. در این بخش نام پروژه‌هایی دیده می‌شود که اخیرا روی آنها کار کرده‌اید؛ بنابراین به راحتی به آنها دسترسی دارید. بخش Start دارای گزینه‌های زیر است:

- **New Project**: ایجاد پروژه جدید
- **Open Project**: باز کردن پروژه موجود
- **Connected to team Foundation Server**: اتصال به سرور برای انجام پروژه‌هایی که به صورت گروهی نوشته می‌شود.

برای نوشتن اولین برنامه در این محیط New Project را انتخاب کنید. کادر محاوره‌ای آن باز می‌شود(شکل ۳-۱۱). در پنل سمت چپ بخش Installed→Templates نام زبان‌های برنامه‌نویسی دیده می‌شود که این برنامه از آنها پشتیبانی می‌کند. با انتخاب هر یک از آنها در پنل وسط قالب‌های آماده از انواع برنامه‌های آن زبان نمایش داده می‌شود. در نسخه Express از ویژوال استودیو برای زبان C#، چهار نوع قالب آماده وجود دارد. انواع برنامه‌های کاربردی قابل تولید به وسیله‌ی سی شارپ در ویژوال استودیو عبارتند از:

- **Console application**: پروژه‌های Console application ساده‌ترین نوع پروژه هستند که از طریق خط فرمان اجرا می‌شوند و ورودی‌های آن از طریق خط فرمان دریافت و خروجی‌های آن نیز در همان محیط نمایش داده می‌شود. این برنامه‌ها مبتنی بر متن هستند.

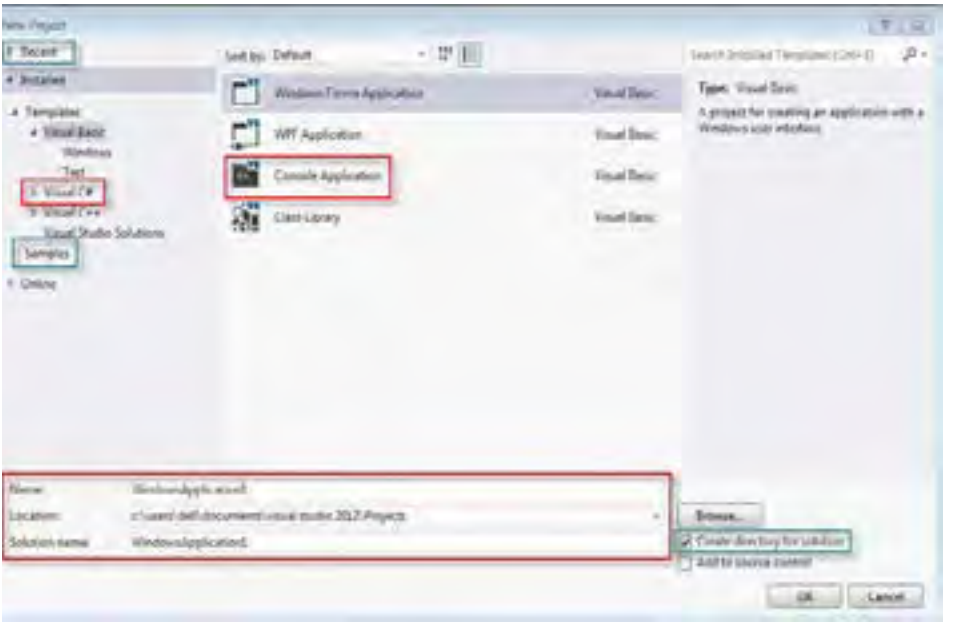
- **Windows forms application**: پروژه‌های Windows forms application مبتنی بر رابط گرافیکی (GUI) هستند و امکان استفاده از امکانات برنامه ویژوال استودیو مانند منوها، دکمه‌ها، فرم‌های ویندوز و کنترل‌های مختلف وکادرهای محاوره رابط کاربر را فراهم می‌کند.

- **Class Library**: پروژه‌های Class Library برای ایجاد کتابخانه‌ها (فایل‌هایی با پسوند dll) به کار می‌روند.

- **WPF application**[۳]: پروژه‌های WPF application به صورت زیرمجموعه‌ای از کلاس‌ها در مجموعه کلاس‌های دات نت هستند که اکثرا در فضای نام «.System Windows» قرارگرفته‌اند. پیش از WPF، ساختن واسط کاربری برای برنامه‌های تحت ویندوز نیازمند به استفاده از فناوری‌های مختلفی مانند GDI+، Windows Forms و بسیاری از فناوری‌های دیگر بود. WPF کارکردهای فناوری‌های مختلف را در یک فناوری کنار هم گردآورده است، تا ایجاد واسط‌های کاربر پیشرفته آسان‌تر گردد. از نظر ظاهری مانند برنامه‌های Windows forms است ودارای امکانات بیشتر از جمله محیط طراحی و ویراستیار XAML[۴] است. این نوع پروژه‌ها در ساخت browserها کاربرد دارند.

۳ Windows Presentation Foundation

۴ Extensible Application Markup Language



شکل ۳-۱۱ : کادر محاوره‌ای New Project

از آنجا که ساده‌ترین نوع پروژه Console application است و نیاز به کنترل یا ابزار خاصی ندارد و مبتنی بر متن است، در کتاب برنامه‌سازی ۱ روی کنسول کار می‌کنیم.

ویژوال استودیو برای مدیریت کارآمد اجزای برنامه شامل منابع، اتصال داده‌ها، پوشه‌ها، فایل‌ها و....دو نوع دسته‌بندی[۵] دارد .این دسته‌بندی‌ها Solution و Project نامیده می‌شوند. پنجره Solution Explorer در ویژوال استودیو برای مشاهده، مدیریت پروژه‌ها و Solutionها و موارد مرتبط با آنها استفاده می‌شود.

Solution شامل مواردی است که شما به منظور ایجاد برنامه‌های کاربردی به آنها نیاز دارید. Solution ساختاری برای سازماندهی پروژه‌ها است و می‌تواند شامل یک یا چند پروژه، به علاوه فایل‌ها و متاداده‌هایی باشد که به تعریف Solution کمک می‌کند. ویژوال استودیو به صورت خودکار هنگام ایجاد پروژه جدید یک Solution ایجاد می‌کند و مشخصات و تعاریف آن را در دو فایل با نام Solution و با پسوند sln و suo[۶] قرار می‌دهد. فایل Solution با پسوند sln فایلی مبتنی بر متن است و شامل متاداده‌هایی است که Solution را تعریف می‌کند. از جمله :

- پروژه‌ها که با Solution در ارتباط هستند.

۵ container

۶ Solution User Options

- مواردی که با یک پروژه خاص مرتبط نیستند.
- تنظیمات مربوط به build کردن پروژه (هنگام کامپایل و ساخت خروجی)

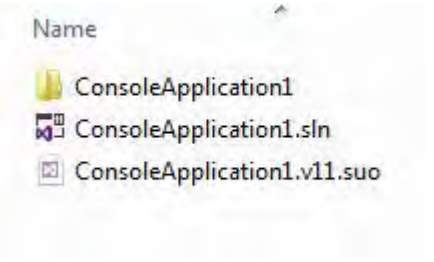
فایل با پسوند suo فایلی باینری و به صورت مخفی است و متاداده ذخیره شده در آن شامل ویژگی‌هایی است در ارتباط با ساختSolution و مجموعه ویژگی‌های مربوط به سفارشی کردن محیط برنامه‌نویسی IDE زمانی که Solution فعال است. به عنوان مثال، Solution Explorer پوشه‌ها و فایل‌های متفاوتی را با توجه به نوع پروژه نمایش می‌دهد و یا در جعبه ابزار، کنترل‌ها و ابزار در دسترس، با توجه به نوع پروژه ایجاد شده متفاوت هستند.

پروژه برای مدیریت منطقی، ساخت و اشکال‌یابی اجزای یک برنامه کاربردی به کار می‌رود. خروجی پروژه فایل اجرایی (exe) یا فایل کتابخانه‌ای (dll) یا یک ماژول برای پروژه‌های دیگر است.

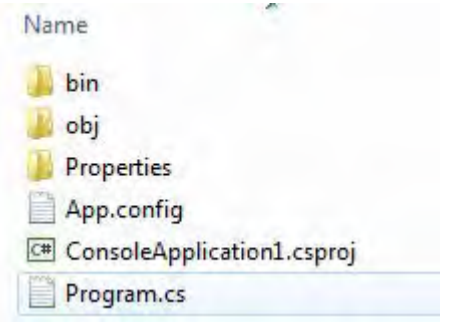
گزینه Recent در پنل سمت چپ کادر محاوره‌ای New Project، نوعی از پروژه را نشان می‌دهد که اخیرا استفاده شده است و گزینه samples در بخش Installed برای شما امکان دانلود مثالهایی از اینترنت را فراهم می‌کند.

در کادر محاوره‌ای New Project از پنل سمت چپ visual C# و از پنل میانی Console Application را انتخاب کرده و در قسمت name نام پروژه را به دلخواه وارد کنید. پیش‌فرض به صورت ConsoleApplicationN است که به جای N یک عدد قرار می‌گیرد. شما می‌توانید برای پروژه نام دلخواه خود را به جای آن قرار دهید. هرنامی که در این قسمت قرار دهید، در قسمت Solution name نیز وارد می‌شود ولی اگر گزینه create directory for solution دارای تیک باشد، شما می‌توانید نام متفاوتی برای Solution قرار دهید. در بخش location محل ذخیره‌سازی پروژه را مشخص کنید و روی دکمه OK کلیک کنید تا الگوی اولین برنامه شما ایجاد شود.

اگر گزینه create directory for solution دارای تیک باشد، پوشه‌ای با نام solution ایجاد می‌شود که شامل فایل‌های Solution (فایل‌هایی با پسوند sln, suo)و پوشه‌ای با نام پروژه است. در پوشه‌ای که با نام پروژه است، فایل‌های مربوط به پروژه قرار دارد. اگر نام پروژه و solution را ConsoleApplication1 قرار داده باشید ،پوشه و فایل‌هایی مطابق شکل ۳-۱۲ در محل تعیین شده در بخش location ایجاد می‌شود که شامل دو فایل solution است. محتوای پوشه ConsoleApplication1 (شکل۳-۱۳) دارای فایلی با پسوند csproj برای پروژه است و فایلی با پسوند cs که کدهای برنامه در آن قرار دارد.



شکل ۳-۱۲ : محتوای پوشه Solution



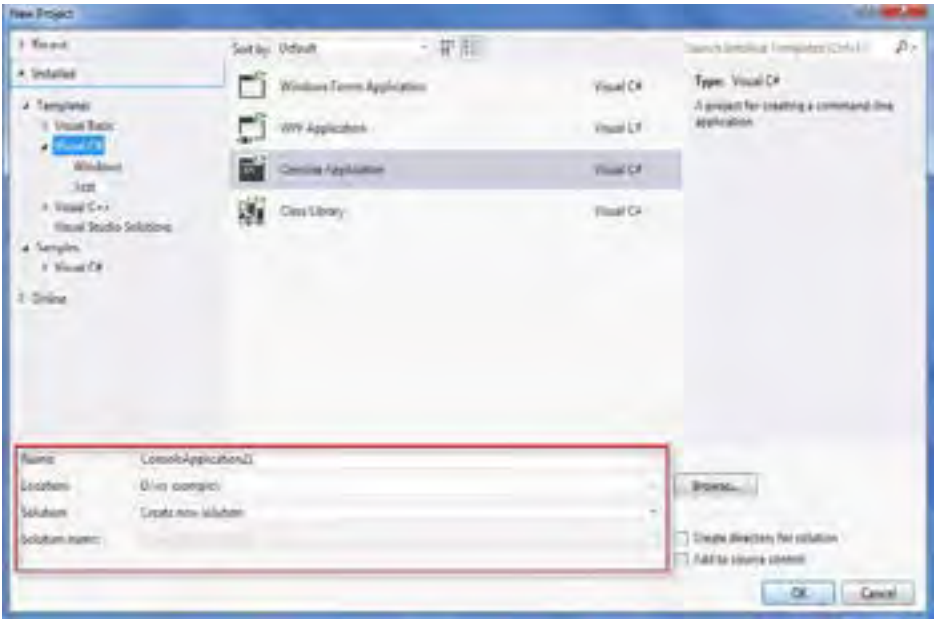
شکل۳-۱۳ : محتوای پوشه پروژه(ConsoleApplication۱)

بنابراین سه نوع از فایل‌های ایجاد شده به وسیله‌ی ویژوال استودیو عبارتند از :

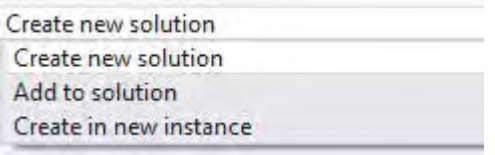
- فایل‌هایی با پسوند sln که معرفSolution ها هستند.
- فایل‌هایی با پسوند csproj که فایل پروژه‌های C# هستند.
- فایل‌هایی با پسوند cs که فایل کدهای نوشته شده به زبانC# هستند.

اگر در حالی که یک solution باز است، بخواهید پروژه جدید ایجاد کنید، از منوی File گزینه New project را انتخاب کنید، کادر محاوره‌ای New Project باز می‌شود(شکل ۱۴-۳). قسمت مشخص شده در شکل۳-۱۴ را با شکل۳-۱۱ (کادر محاوره‌ای New Project زمانی که هیچ solutionی باز نیست) مقایسه کنید. در کادر محاوره‌ای New Project زمانی که یک Solution باز داریم، کادر لیست Solution اضافه شده است که دارای ۳ گزینه زیر است:

- **Create new solution**: solution جدید ایجاد کرده، پروژه را درآن ایجاد می‌کند؛ بنابراین قسمت وارد کردن نام solution فعال است. Solution قبلی بسته شده، solution جدید در ویژوال استودیو دیده می‌شود.
- **Add to solution**: تنها پروژه را ایجاد کرده، آن را به Solution موجود اضافه می‌کند؛ بنابراین قسمت نام Solution غیرفعال است.
- **Create in new instance**: مانند گزینه Create new solution عمل کرده و solution جدید ایجاد کرده، پروژه را در آن ایجاد می‌کند. با این تفاوت که solution جدید را در پنجره ویژوال استودیو دیگری نمایش می‌دهد و solution قبلی بسته نمی‌شود. در این حالت نیز مانند حالت Create new solution قسمت وارد کردن نام solution فعال است.



شکل۳-۱۴ : کادر محاوره‌ای New Project در زمان باز بودن یک solution

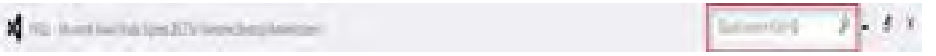


شکل ۳-۱۵ : گزینه‌های لیست Solution در کادر New Project

۳-۴-آشنایی با محیط ویژوال استودیو

قسمت‌های مختلف محیط IDE ویژوال استودیو و راه دسترسی به پنجره‌ها در زیر آمده است.

- نوارمنو و نوار ابزار
- پنجره ویرایشگر برنامه
- پنجره Solution Explorer: کلید دسترسی آن CTRL+W,S است .
- پنجره لیست خطاها: کلید دسترسی آن CTRL+W,E است.



شکل ۳-۱۶ : نوار عنوان برنامه Visual Studio

شکل ۱۶-۳ نوار عنوان برنامه ویژوال استودیو را نشان می‌دهد. در قسمت نوار عنوان کادر Quick Launch قرار دارد که عملیات جستجو در منوها و تنظیمات ویژوال استودیو را انجام داده و فهرستی از موارد پیدا شده را نمایش می‌دهد؛ بنابراین هرگاه محل گزینه‌ها را در منوها و یا تنظیمات ویژوال استودیو فراموش کردید، می‌توانید از این روش برای دسترسی به آن استفاده کنید.

۳-۴-۱- نوارمنو

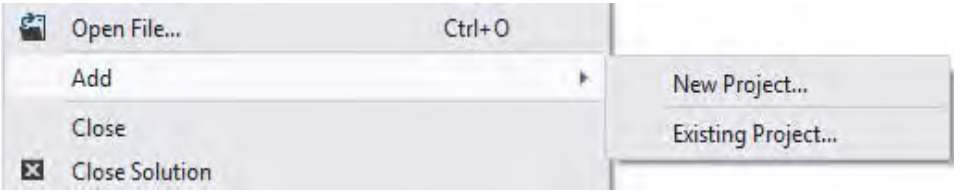
شکل ۳-۱۷ : نوار منو

در ادامه برخی از گزینه‌های منوها را که کاربرد بیشتری دارند، بیان می‌کنیم.

- **منـــوی File**: ایـن منـو شـامل گزینه‌هایـی بـرای ایجـاد و ذخیره‌سـازی، بـاز کـردن فایـل، پـروژه و Solution است(شـکل۳-۱۸ و۳-۱۹). گزینه‌هـای ایـن منـو عبارتنـد از:
  - **New Project**: ایجاد پروژه جدید
  - **Open Project**: باز کردن پروژه موجود
  - **Add**: بـرای اضافـه کـردن پـروژه بـه Solution ی کـه بـاز اسـت .ایـن منـو خـود دارای دو گزینـه New Project بـرای اضافـه کـردن پـروژه جدیـد و گزینـه Existing Project بـرای اضافـه کـردن پـروژه‌ای کـه قبـلا ایجـاد شـده بـه Solution (شـکل۳-۲۰)

شکل ۳-۱۹ : منوی File در حالت کلی

شکل ۳-۱۸ : منوی File در زمان انتخاب فایل Program.cs در Solution Explorer



شکل ۳-۲۰ : آیتم **Add** از منوی **File**

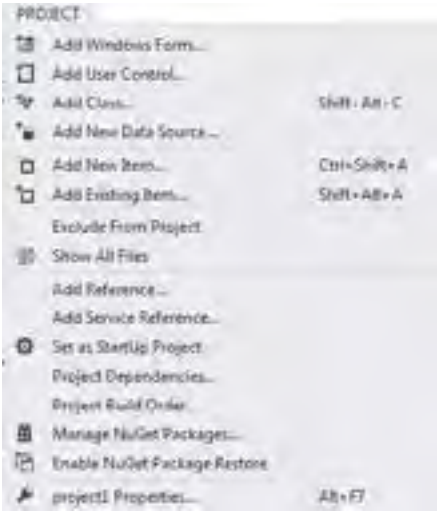
- **Close**: بستن فایل فعال
- **Close Solution**: بستن Solution به شکل کامل
- **Save Selected Item**: ذخیره فایل فعال و اگر فایلی فعال نباشد، ذخیره فایلی که در پنجره Solution Explorer انتخاب شده است که در این حالت به جای Selected Item نام فایل انتخاب شده در منو دیده می‌شود.(شکل ۳-۱۸ و۳-۱۹ را با هم مقایسه کنید.)

- **Save Output AS..**: مانند گزینه Save As در برنامه‌های ویراستیار، فایل فعال یا انتخاب شده را با نام جدید ذخیره می‌کند.
- **Save All**: ذخیره تمام فایل‌های موجود در Solution
- **Recent Files**: فهرستی از فایل‌های که اخیرا باز شده را برای دسترسی سریع‌تر به آنها نشان می‌دهد.
- **Recent Projects and Solutions**: فهرستی از پروژه‌ها و Solution هایی که اخیرا باز شده را برای دسترسی سریع‌تر به آنها نشان می‌دهد.

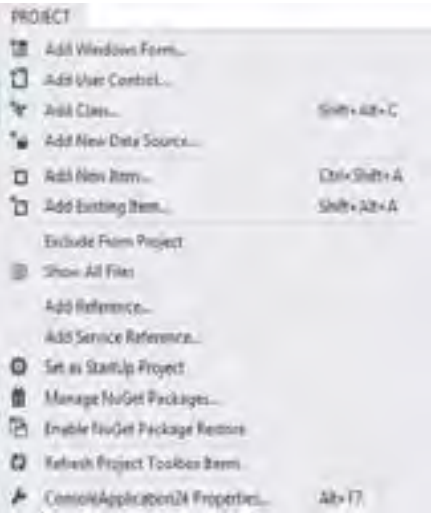
- منوی **View**: این منو برای دسترسی به پنجره‌های محیط ویژوال استودیو به کار می‌رود. شما می‌توانید هر پنجره‌ای را که نیاز دارید، از طریق این منو و یا با استفاده از کلیدهای میانبر آنها (شکل ۳-۲۱) بازکنید.

شکل ۳-۲۱ : منوی view

- **منـــوی project**: ایـــن منـــو بـــرای اضافـــه کـــردن انـــواع فایل‌هـــا بـــه پـــروژه، تعییـــن پـــروژه startup زمانـــی کـــهSolution دارای چنـــد پـــروژه باشـــد و ... بـــه کار مـــی‌رود.

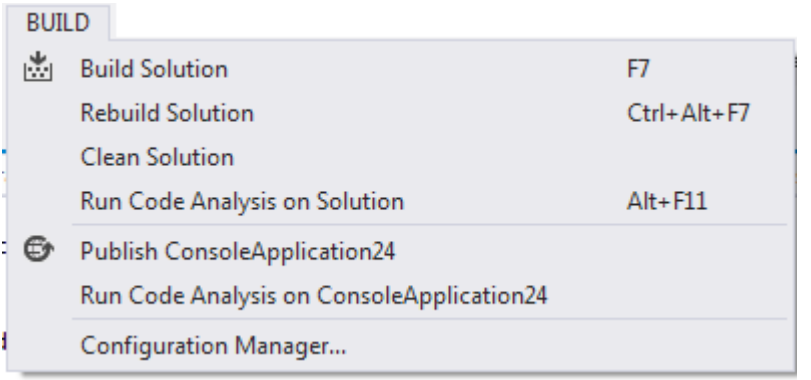


شکل ۳-۲۲ : منوی Project زمانی که Solution دارای یک پروژه است.



شکل۳-۲۳ : منوی project زمانی که Solution دارای چند پروژه باشد.

- منـــوی Build: بـــرای ســـاخت خروجی‌هـــای Solution اســـتفاده می‌شـــود کـــه بـــا توجـــه بـــه اینکـــه تنظیـــم ترجمـــه در حالـــت Debug یـــا Release باشـــد، فایل‌هـــای خروجـــی را در پوشـــه‌ای بـــا نـــام Debug یـــا Release ذخیـــره می‌کنـــد کـــه در مســـیر " bin\نـــام پـــروژه " قـــرار دارد(شـــکل۳-۲۴).

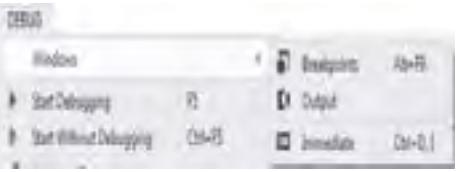


شکل۳-۲۴ : منوی Build

گزینه‌های منوی Build عبارتند از:

- **Build Solution**: ساخت فایل‌های خروجی Solution
- **Rebuild Solution**: فایل‌های خروجی که قبلا ساخته شده را حذف کرده، دوباره فایل‌های خروجی را می‌سازد.
- **Clean Solution**: فایل‌های خروجی که قبلا ساخته شده را حذف می کند.
- **Publish Project name**: (به جای Project name نام پروژه قرار دارد) برای پروژه بسته نرم‌افزاری قابل نصب تولید می‌کند که دارای فایل Setup برای نصب بسته روی سیستم و فایل‌های Uninstall برای حذف بسته از سیستم است.

- **منوی Debug**: این منو برای ترجمه و اشکال‌یابی فایل‌های کد، پروژه و استفاده از امکانات اشکال‌یابی مانند نقطه قطع به کار می‌رود(شکل۳-۲۶).



شکل۳-۲۵ : گزینه‌های ایتم window در منوی debug(فهرست پنجره‌های مربوط به اشکال یابی)

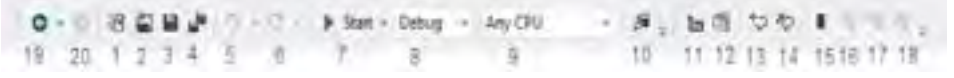
شکل۳-۲۶ : منوی debug

گزینه‌های منو Debug عبارتند از:

- **Windows**: برای باز کردن پنجره‌های breakpoint، Output و Immediate در زمان اشکال‌یابی برنامه و اجرای خط به خط برنامه
- **Start Debugging(F5)**: ترجمه برنامه و اجرا در صورت نداشتن خطا
- **Start Without Debugging(Ctrl+F5)**: اجرای برنامه بدون امکانات اشکال‌یابی[۷]
- گزینه‌های step Into، Step Over، Toggle Breakpoint، New Breakpoint و Delete All برای اشکال‌یابی برنامه استفاده می‌شود.

در همین فصل به تفاوت‌های انواع ترجمه و اجرای برنامه خواهیم پرداخت.

۳-۴-۲- نوار ابزار



شکل ۳-۲۷ : نوار ابزار

برخی از ابزارهای موجود در نوار ابزار ویژوال استودیو در شکل۳-۲۷ نشان داده شده، شماره‌گذاری شده‌اند. در ادامه با توجه به شماره‌ها شرح مختصری از

۷ اشکال‌یابی (Debugging) امکاناتی برای بررسی نحوه اجرا و یافتن اشکالات در حین اجرا در اختیار می‌گذارد.مانند Breakpoint، کلاس Debug و ...

هریـک آمـده اسـت.

۱ **New Project(ctrl+shift+N)**: ایجاد پروژه جدید

۲ **Open File (Ctrl+O)**: باز کردن فایل

۳ **Save filename(Ctrl+S)**: ذخیره کردن فایل

۴ **Save All(Ctrl+shift+S)** :ذخیره کردن تمامی فایل‌ها

۵ **Undo(Ctrl+Z)**: لغو آخرین عمل

۶ **Redo(Ctrl+Y)**: بازگرداندن آخرین عمل

۷ **Start**: شروع ترجمه برنامه

۸ **Solution Configurations**: لیسـتی بـرای تعییـن وضعیـت ترجمـه کـه می‌توانـد Debug، Release و یـا Configuration Manager باشـد.

۹ **Solution Platform**: تعییـن نـوع cpu ی کـه برنامـه بتوانـد روی آن اجـرا شـود. در حالـت Any CPU روی تمـام CPUهـا اجـرا می‌شـود.

۱۰ **Find In Files(Ctrl+shift+F)**: جستجو در تمامی پرونده‌ها و فایل‌ها

۱۱ **Display Quick info(Ctrl+K,I)** :نمایـش اطلاعـات درمـورد قسـمتی کـه مکان‌نمـا روی آن قـرار دارد. در ویـژوال اسـتودیو وقتـی مـاوس روی هـر قسـمت از برنامـه قـرار گیـرد، کادری بـاز شـده، در مـورد متـد، کلاس، متغیـر ،... و هرچیـزی کـه مـاوس روی آن قـرار دارد توضیحاتـی ظاهـر می‌شـود. بـه ایـن کادر Quick Info می‌گویند(شـکل ۳-۲۸).

```
        static void Main(string[] args)
        {
            Console.ForegroundColor = System.ConsoleColor.Blue;
            Console.
        }
    }
}
```

ConsoleColor Console.ForegroundColor
Gets or sets the foreground color of the console.

Exceptions:
System.ArgumentException
System.Security.SecurityException
System.IO.IOException

شکل۳-۲۸ : Quick info برای مشخصه ForegroundColor در کلاس Console

۱۲ **Toggles between suggestion and standard completion modes. (Ctrl +Space)**: در ویژوال استودیو امکان تکمیل نام کلمات کلیدی، متدها، مشخصات، نام کلاس، فضای نام و .... به صورت خودکار وجود دارد که به آن intelliSense گفته می‌شود و این ویژگی دارای دوحالت است.

۱. **تکمیل کردن (completion که حالت پیش‌فرض است)**: در لیست نمایش داده شده، نام پیشنهادی دارای کادر است و اولین عنصر لیست همان کلمه‌ای است که تایپ می‌کنید(شکل ۳۰-۳).

۲. **پیشنهاد کردن (suggestion)**: نام کامل در لیست نمایش داده شده به صورت highlight دیده می‌شود(شکل ۲۹-۳).

شکل۳-۲۹ : حالت پیشنهادی

شکل۳-۳۰ : حالت تکمیلی

۱۳ **Comment out the selected lines(Ctrl+E,C)**: با کلیک این نشانه ابتدای خطی که مکان‌نما روی آن قرار دارد و یا ابتدای خطوطی که در حالت انتخاب قرار دارند، علامت // قرار می‌گیرد، بنابراین آن خطوط به توضیحات تبدیل شده، ترجمه و اجرا نخواهد شد.

۱۴ **UnComment the selected lines(Ctrl+E,U)**: حذف علامت // از ابتدای خط یا خطوط انتخاب شده که سبب می‌شود از حالت توضیحات خارج شده، ترجمه و اجرا شوند.

۱۵ **Toggle a bookmark on the current line(Ctrl+B,T)**: برای قرار دادن یا برداشتن Bookmark از روی یک خط

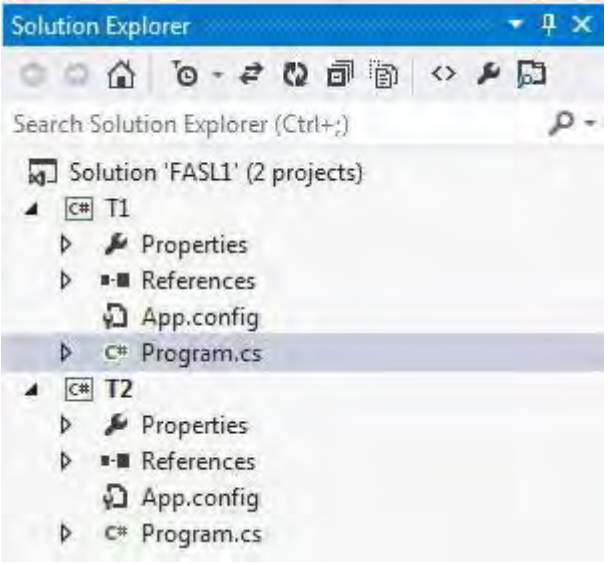
۱۶ **Move the caret to the previous bookmark(Ctrl+B,P)**: حرکت از Bookmarkجاری و رفتن به Bookmark قبلی

۱۷ **Move the caret to the next bookmark(F2)**: حرکت از Bookmak جاری و رفتن به Bookmark بعدی

۱۸ **Clear all bookmarks in all files.(Ctrl+B,C)**: حذف تمام Bookmarkها در تمام فایل‌های Solution

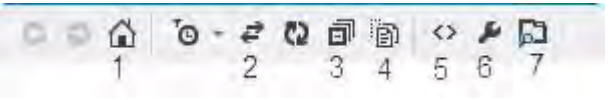
۳-۴-۳- پنجره Solution Explorer

پنجره Solution Explorer به صورت پیش‌فرض در سمت راست پنجره ویژوال استودیو قرار دارد و فهرستی از پروژه‌های Solution جاری و فایل‌های موجود در پروژه‌ها را نمایش می‌دهد(شکل۳-۳۱). به وسیله‌ی این پنجره به راحتی به تمام فایل‌ها و اجزای Solution دسترسی داریم.



شکل۳-۳۱ : پنجره Solution Explorer

در این پنجره با دوبار کلیک کردن روی نام فایل، محتویات آن فایل در محیط ویژوال استودیو نمایش داده خواهد شد. اگر گزینه Preview Selected Items (نشانه شماره ۷ در شکل۳-۳۲) در پنجره Solution Explorer در حالت انتخاب باشد، با یک بار کلیک بر روی فایل، محتویات آن نمایش داده خواهد شد.



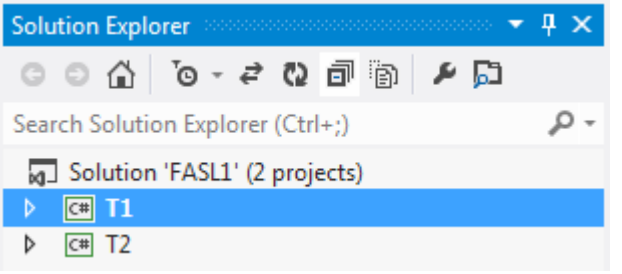
شکل۳-۳۲ : نوار ابزار پنجره Solution Explorer

برخی از نشانه‌های نوار ابزار پنجره Solution Explorer را که در شکل ۳-۳۲ شماره‌گذاری شده است و کاربرد بیشتری دارد در ادامه آمده است.

۱- **Home**: اگر محتوای این پنجره را با تغییر محدوده نمایش ساختار درختی تغییر دهیم، با کلیک این نشانه دوباره ساختار درختی اصلی نمایش داده می‌شود.

۲- **Sync with Active Document(Ctrl+[,S)**: فایل فعال را در ساختار درختی به صورت انتخاب در می‌آورد. با استفاده از این گزینه می‌توانید محل فایلی که روی آن کار می‌کنید را در پروژه مشخص کنید.

۳- **Collapse All**: در پنجره فقط نام Solution و پروژه‌های آن را نشان می‌دهد. درواقع ساختاردرختی را جمع می‌کند.(شکل۳-۳۳)

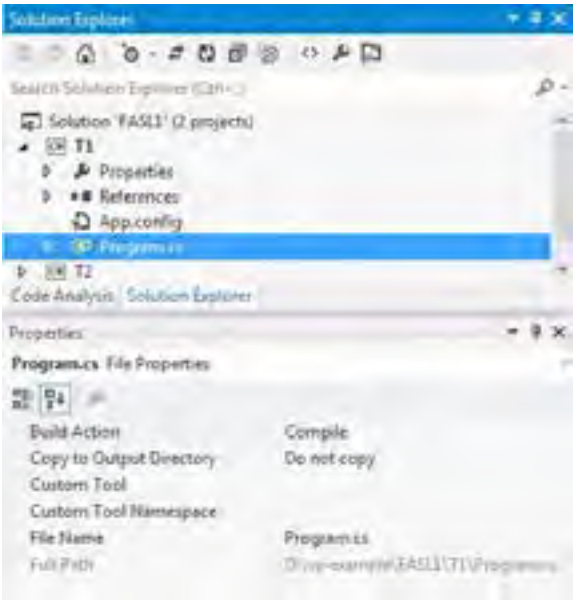


شکل۳-۳۳ : ساختار درختی جمع شده (Collapse All)

۴- **Show All Files**: تمام فایل‌های پوشه‌ها را نشان می‌دهد. در واقع ساختار درختی را باز می‌کند.

۵- **View code**: برای مشاهده کد فایل انتخاب شده، می‌توان از آن استفاده کرد. به جای دوبار کلیک روی فایل، می‌توان فایل را انتخاب کرد و سپس این نشانه را کلیک کرد که محتوای فایل نمایش داده شود.

۶- **Properties**: ویژگی‌ها و خصوصیات آیتمی که انتخاب شده را نشان می‌دهد. برای مشاهده محلی که فایل ذخیره شده، می‌توانید از این نشانه استفاده کنید و ویژگی‌های فایل را در پنجره properties مشاهده کنید. در شکل۳-۳۴ مسیر فایل program.cs در پنجره Properties مشاهده می‌شود.



شکل۳-۳۴ : استفاده از نشانه properties پنجره solution explorer

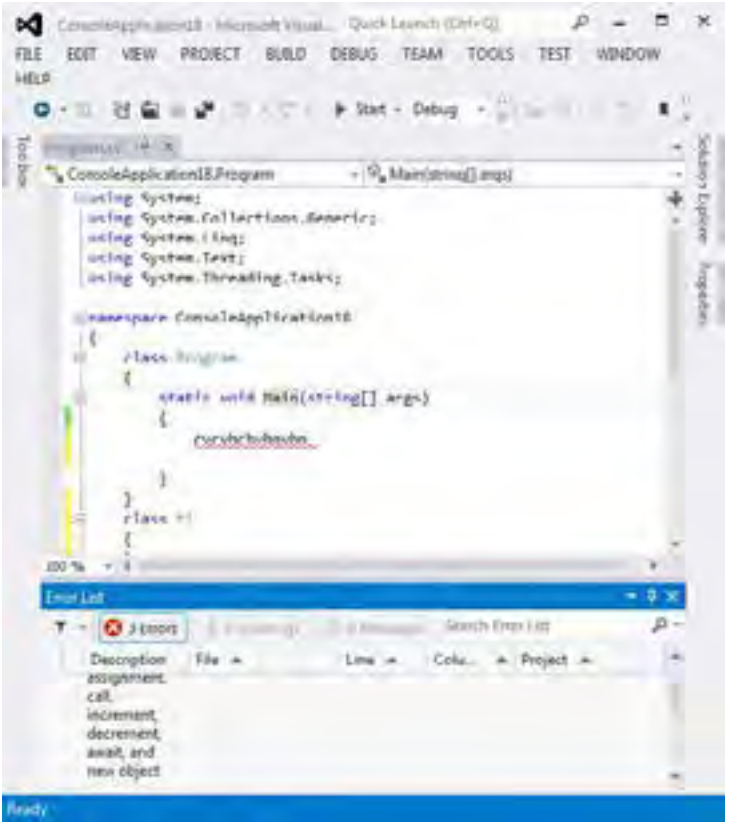
۷- **Preview Selected Items**: اگر این گزینه در حالت انتخاب باشد، با یک بار کلیک بر روی فایل، محتویات آن نمایش داده خواهد شد.

اگر پنجره Solution Explorer را مشاهده نمی کنید، از منوی View گزینه Solution Explorerرا انتخاب کنید تا این پنجره نمایش داده شود. همچنین برای استفاده بهینه از فضای ویژوال استودیو، بهتر است قابلیت Auto Hide این پنجره را فعال کنید.

شکل۳-۳۵ : نشانه Auto Hide پنجره Solution Explor

علامت Auto Hide که شکل آیکن آن به صورت پونز است .در نوار عنوان پنجره‌ها سمت راست قرار دارد و با استفاده از آن می‌توان پنجره را به صورت خودکار مخفی کرد. اگر علامت به شکل باشد، پنجره مخفی می‌شود و تنها نام آن در یکی از کناره‌ها دیده می‌شود و با کلیک نام آن، پنجره باز می‌شود و پس از کلیک خارج از پنجره دوباره مخفی می‌شود و اگر علامت به شکل پونز باشد، پنجره در جای خود ثابت می‌ماند. در شکل۳-۳۶ پنجره

Solution و Properties مخفی شده و عنوان آنها در کناره سمت راست دیده می‌شود و پنجره لیست خطاها در جایش ثابت شده است و نشانه Auto Hide آن به صورت پونز است. با کلیک کردن بر روی عنوان Solution Explorer یا Properties پنجره آن نمایش داده می‌شود و با کلیک کردن جایی خارج از آن، دوباره ناپدید می‌شود. مخفی کردن پنجره‌ها فضای بیشتری برای پنجره‌های دیگر فراهم می‌کند.



شکل۳-۳۶ : Auto Hide شدن پنجره‌های Solution و Properties

۳-۴-۴- پنجره لیست خطاها(Error List)

در صورتی که برنامه دارای خطا باشد، فهرست خطاها شامل شرح خطا، فایلی که دارای خطا است، شماره خط، شماره ستون و نام پروژه‌ای که خطا در آن رخ داده، در پنجره لیست خطاها مشاهده می‌شود(شکل۳-۳۷). با دوبار کلیک روی هر خطا مکان‌نما به خطی از کد منتقل می‌شود که خطا در آن رخ داده‌است.

شکل۳-۳۷ : پنجره لیست خطاها

۳-۵- ترجمه و اجرای برنامه

بعد از نوشتن برنامه، باید آن را ترجمه و اجرا کنید. برای اجرا کردن برنامه می‌توانید از سه روش استفاده کنید.

۱ - کلیک بر روی دکمه Start که در نوار ابزار ویژوال استودیو قرار دارد.

۲ - انتخاب گزینه Start Debugging از منوی Debug

۳ - کلید میانبر F5 که این راه سریعترین روش برای اجرا می‌باشد.

علاوه بر گزینه start Debugging گزینه Start without debugging(Ctrl+F5) برای ترجمه و اجرا و گزینه Build(F7) برای ترجمه و تولید خروجی در ویژوال استودیو وجود دارد. مقایسه این سه گزینه در ادامه آمده است.

Build(F7): کد برنامه را که کد متنی است، به کد IL مشترک بین تمام زبان‌های Net. تبدیل می‌کند و خروجی را بدون کدهای اشکال‌یابی ایجاد می‌کند.

Start(F5): شروع به ترجمه می‌کند و کدهای مربوط به اشکال‌یابی را هم ترجمه می‌کند؛ بنابراین در این حالت می‌توان برنامه را خط به خط اجرا کرد، نقطه قطع قرار داد و یا هر کاری که برای رفع عیب برنامه لازم است را انجام داد.[۱]

Start without debugging(Ctrl+F5): شروع به ترجمه می‌کند و از کدهای مربوط به اشکال‌یابی صرف نظر می‌کند؛ بنابراین در این حالت نمی‌توان برنامه را خط به خط اجرا کرد و عملیات اشکال‌یابی را نمی‌توان انجام داد ولی خروجی به صورت قابل توجه سریع‌تر و بهینه می‌شود.

برای بار اول اجرای برنامه، هر کدام از کلیدهای میانبر F5 و Ctrl+F5 را انتخاب

۱ فایل‌های pdb تنها در حالت debug ایجاد می‌شوند(فایل‌های شامل اطلاعات debugging) و با دیکامپایلرها کد و توضیحات برنامه را می‌توان به دست آورد.

کنید، ابتدا عمل ترجمه و تبدیل به زبان مشترک .Net (IL) انجام می‌شود؛ اما اگر از کلید میانبر F7 استفاده کنید، فقط برنامه ترجمه شده، خطاهای کامپایلر مشاهده می‌شود و اگر برنامه خطایی نداشته باشد، خروجی‌ها (از جمله فایل exe) ساخته می‌شود؛ ولی اجرا نمی‌شود.

فشردن کلیدهای Ctrl+F5 باعث می‌شود که فایل exe خارج از محیط و کنترل ویژوال استودیو اجرا شود. این عمل مانند این است که شما روی فایل exe برنامه دابل کلیک کرده باشید. در این صورت ویژوال استودیو و ابزارهای اشکال‌یابی آن هیچ کنترلی در اجرا برنامه ندارند. اگر در محیط کنسول برنامه نوشته شده باشد، در این حالت پیام press any key to continue .... در پایان کار نمایش داده خواهد شد.

وقتی شما با کلید میانبر F5 پروژه را اجرا کنید، ویژوال استودیو و ابزارهای اشکال‌یابی آن تمام مراحل اجرای دستورات برنامه را کنترل می‌کنند، که چه متغیری داریم و چه مقداری دارد، کدام تابع و چه زمانی اجرا شده است و بعد مقدار متغیرها چه تغییری کرده است و... پس در صورت بروز خطا این اطلاعات می‌تواند در رفع خطا کمک کند.

واژگان

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	Integrated Development Environment	محیط یکپارچه توسعه
۲	Debug	اشکال یابی
۳	Error	خطا
۴	Warning	هشدار

پرسش و پاسخ

۱- دو گزینه add to source control و create directory for solutions چه تفاوتی با هم دارند؟

گزینه‌ی اول در برنامه‌نویسی تیمی کاربرد دارد، تا همه فایل‌ها طبق ورژن در سرور مخزن تحت مدیریت باشد.

گزینه دوم به ازای هر Solution یک پوشه می‌سازد و پروژه‌های آن Solution را در پوشه‌های مجزایی درون آن پوشه قرار می‌دهد.

۲- تفاوت Solution و Project در چیست؟

بالاترین سطح دسته‌بندی در محیط ویژوال استودیو دات نت را Solution می‌گویند. یک Solution می‌تواند شامل چند Project مختلف باشد.

وظیفه Solution نگهداری اطلاعات روابط بین پروژه هاست که یک فایل با پسوند sln است و تاثیری در خروجی پروژه نخواهد داشت.

در مقابل Project یا همان پروژه‌ها با توجه به نوعشان دارای خروجی مستقیم هستند. به عنوان مثال Console Application ها دارای خروجی با پسوندexe به معنای executable یا همان فایل‌های قابل اجرا هستند.

۳-همه پروژه‌های یک Solution به یک زبان هستند یا می‌توانند به زبان‌های متفاوت باشند؟

می‌توانند به زبان‌های متفاوتی از ویژوال استودیو دات نت باشد. البته Solution می‌تواند هیچ پروژه‌ای هم نداشته باشد.

۴- اگر دانش‌آموزی بپرسد کار String args چیست؟ چگونه در حد دانش وی باید توضیح داده شود؟

از این پارامتر، برای نوشتن برنامه‌هایی استفاده می‌شود که از طریق خط فرمان پارامتر می‌گیرند. در این حالت متد Main باید آرایه‌ای از پارمترهای رشته‌ای را به برنامه ارسال کند. دستور اجرای برنامه‌ای که پارامتر دریافت می‌کند، به شکل زیر است:

```
MyApp.exe   Arg1   Arg2   Arg3....
```

۵- تفاوت F6 و F5 و Ctrl+F5 در C# چیست ؟

با Ctrl+F5 برنامه نوشته شده کامپایل می‌شود و در صورتی‌که برنامه خطایی داشته باشد، در آخر گزارش می‌دهد و اگر خطا نداشته باشد، فایل اجرایی ساخته شده، اجرا می‌شود. برنامه اگر برای محیط Console نوشته شده باشد، در محیط Command Prompt بدون توجه به Breakpoint اجرا و در انتهای اجرای برنامه پیغام press any key to continue نمایش داده خواهد شد.

با F5 برنامه کامپایل می‌شود و در صورتی‌که برنامه خطایی نداشته باشد، فایل اجرایی هم ساخته خواهد شد و همچنین قابلیت اجرای خط به خط برای خطایابی وجود دارد و فایل اجرایی شامل اطلاعات اضافی برای خطایابی خواهد بود و می‌توان با دیکامپایلرها سورس و توضیحات برنامه را بدست آورد.

با استفاده از F6 برنامه نهایی ساخته می‌شود، فایل خروجی بهینه و مطالب اضافی همراه فایل نخواهد بود و امکان خطایابی هم وجود ندارد و برنامه بعد از ساخته شدن نیز بصورت خودکار اجرا نخواهدشد.

واحد کار

واحد کار شماره: ۳-۱	فصل ۳ - صفحه ۴۱ تا ۵۳	مدت آموزش: ۹۰ دقیقه	روش تدریس پیشنهادی: پیش سازمان دهنده- پرسش و پاسخ.
عنوان: آشنایی با ویژوال استودیو		هدف کلی: آموزش اجزای ویژوال استودیو	
اهداف رفتاری: انتظار می‌رود دانش آموز ۱. ویژگی های محیط IDE را بیان کند. ۲. بتواند در نرم افزار VS برنامه از نوع کنسول بنویسد. ۳. از پنجره های محیط VS در برنامه نویسی استفاده کند.		اهداف پنهان درس: هدف ایده آل:	
فرایند تدریس: از هنرجویان بخواهیم که مراحل نوشتن تا اجرای برنامه را بیان کنند و بگویند که در فصل ۲ برای هر مرحله از چه نرم افزاری استفاده کرده اند؟ حال اگر نرم افزاری محیطی فراهم کند که بتوان تمام این مراحل را در آن انجام داد ،کار برنامه نویس را راحت می کند . این محیط را IDE می گویند و VS چنین محیطی را فراهم می کند. اجرای VS و انجام مراحل ایجاد یک پروژه از نوع کنسول را توضیح می دهیم(توضیح صفحه Start Page و انتخاب نوع پروژه، تعیین نام و محل پروژه و توضیح مفهوم solution در VS) معرفی پنجره های VS و کاربرد هریک از آنها شامل : ۱- پنجره Solution Explorer: که نمایش ساختار پروژه و فایلهای آن است. ۲- پنجره ویرایشگر: جلب توجه هنرجو به اینکه با ایجاد پروژه، الگوی اصلی برنامه نوشته شده است. مقایسه الگو با برنامه هایی که در فصل ۲ هنرجو نوشته است و بیان تفاوتها (وجود معرفی فضاهای نام آماده در ابتدا برنامه و فضای نام پروژه که کلاس در آن قرار دارد.). تعریف فضای نام و کاربرد آن برای هنرجو و در نهایت نوشتن یک برنامه مانند نوشتن نام و نام خانوادگی و نام مدرسه روی صفحه نمایش و معرفی ویژگی intellisense در هنگام نوشتن کد، قرار ندادن علامت ; در انتهای یک خط و جلب توجه هنرجو به خط قرمز رسم شده زیر کدی که اشکال دارد.۳- ترجمه با f6 و معرفی پنجره Error List و رفع خطا ۴- در نهایت اجرا با کلید f5 یا دکمه start ۵- بیان تفاوت اجرا با f5 و ctrl+f5 (متد ReadKey را در انتهای کد قرار ندهیم و برنامه را یکبار با f5 و یکبار با ctrl+f5 اجرا کنیم.)			

واحد کار شماره : ۳-۱	فصل ۳ - صفحه ۴۱ تا ۵۳

توصیه های اجرایی:

- از هنرجویان، درخواست شود که چند برنامه دیگر که در فصل ۲ نوشته اند را در اینجا دوباره بنویسند.
- روش اضافه کردن فایلهای برنامه که قبلا نوشته شده را به پنجره Solution explorer نشان دهید.
- از هنرجو بخواهیم پس از حذف معرفی فضاهای نام از ابتدای برنامه، برنامه را اجرا کند و خطاها را بررسی نماید.

ارزشیابی از انتظارات:

۱. پنجره Solution Explorer ساختار و فایلهای پروژه را نمایش می دهد. ص غ
۲. برای ترجمه برنامه از کلید Ctrl+f5 استفاده می کنیم. ص غ
۳. اجرای برنامه با کلید F5 انجام می شود. ص غ
۴. اگربرنامه را با کلید F5 اجرا کنیم، برای توقف برنامه و دیدن خروجی نیاز به متد ReadKey نیست. ص غ
۵. محیط برنامه نویسی که در آن بتوان تمام مراحل .....و ..... و ..... و سرانجام اجرا را در آن انجام داد، محیط IDE می‌گویند.
۶. در Vs لیست خطاها در پنجره ..... دیده می شود.
۷. برای سازماندهی و دسته بندی پروژه های بزرگ از ..... استفاده می شود.

تکالیف و پروژه های پیشنهادی:

۱. برنامه ای بنویسید که چهار عمل اصلی را روی دو عدد ۴۵و ۲۳ انجام داده و نتیجه را با پیام مناسب بنویسد.
۲. برنامه ای بنویسید که نام و سن اعضای گروه شما را بنویسد(هنرجویان به صورت گروههای ۲ نفره کار می کنند.)
۳. برنامه زیر را نوشته و خطاهای آن را رفع کنید.

```
Console.WriteLine(«a+b=»+ 20 +5);
Console.WriteLine(«a+b=»+( 20 +5));
Console.ForegroundColor = ConsoleColor.Black;        Console.Writeline(«a-b=» + 5 * 20);
```

واحد کار

واحد کار شماره: ۳-۲	فصل ۳- صفحه ۴۸ تا ۵۳	مدت آموزش: ۳۰ دقیقه	روش تدریس پیشنهادی: پرسش و پاسخ.
عنوان: برنامه جدید		هدف کلی: نوشتن اولین برنامه در محیط ویژوال استودیو	
اهداف رفتاری: انتظار می رود دانش آموز ۱. طرز نوشتن یک برنامه جدید در محیط ویژوال استودیو را توضیح دهند. ۲. قسمتهای اصلی یک برنامه را شرح دهد. ۳. بتواند اشکالات اولیه را رفع نماید.		اهداف پنهان درس: ساختار کامل یک برنامه را بتواند بطور کامل توضیح داده، نوشته و اجرا نماید. هدف ایده آل:	

فرایند تدریس:

از دانش آموزان می خواهیم یک پروژه جدید به نام testProgram ایجاد نمایند و قطعه برنامه ای بنویسند که نام و نام خانوادگی خود را در خروجی چاپ نمایند.

سپس سوالاتی مطرح می شود که هنرجو با محیط برنامه نویسی و کاربرد قسمتهای مختلف آن آشنا شود. این سوالات عبارتند:

۱- در صورت اشکال در برنامه، در کدام قسمت اشکالات برنامه را می توان مشاهده نمود؟

۲- آیا نام برنامه خود را می توان به Prog1 تغییر داد؟

۳- چگونه می توان خروجی آن را مشاهده کرد؟

۴- آیا تمامی کدها یکرنگ می باشند؟

۵- آیا همیشه لیست هوشمند دستورات باز می شود؟ یا فقط برای کلماتی خاص باز می شود؟

پس از آمادگی ذهنی هنرجویان، به توضیح و نمایش نحوه اشکال گیری و اجرای برنامه در محیط ویژوال استودیو می پردازیم. سپس با دادن برنامه دیگری به دانش آموزان، آنها را به سمت نوشتن و آشنا شدن با محیط برنامه نویسی هدایت می نماییم.

واحد کار شماره : ۳-۲	فصل ۳- صفحه ۴۸ تا ۵۳

توصیه های اجرایی:

- تفاوت کلمات کلیدی با کلمات ساده را عنوان نمایید.
- در ابتدای کار تاکیدی بر استفاده از لیست هوشمند دستورات ویژوال استودیو در هنگام برنامه نویسی داشته باشید که کمتر دچار اشکالات املایی شوند.
- کلیدهای میانبر اجرا و مترجم را (f7 , f5) در هنگام اجرای برنامه به دانش آموزان تاکید نمایید.
- توجه دانش آموزان را به پسوند ذخیره فایل برنامه جلب نمایید.

ارزشیابی از انتظارات:

۱. برای ترجمه برنامه از کلید میانبر f7 استفاده می نماییم. ص غ
۲. برای اجرای صحیح برنامه می توان از کلید میانبر crtl+f6 استفاده نمود. ص غ
۳. برای مشاهده اشکالات یک برنامه می توان پنجره ..... را باز کرد.
۴. هر برنامه در حالت کلی بین دو علامت ..... و ..... قرار می گیرد.({ و })

تکالیف و پروژه های پیشنهادی:

۱. برنامه ای بنویسید که مشخصات هنرستان شما را با رنگ آبی نمایش دهد. برنامه را در محیط ویژوال نوشته و اجرا نماید.
۲. پنجره لیست خطا را توضیح دهید .

نمونه سوال از فصل ۳

درستی یا نادرستی هر عبارت را تعیین کنید. د برای درست و ن برای نادرست.

۱ - پنجره Solution Explorer ساختار پروژه و تمام فایل‌های موجود در آن را نشان می‌دهد.

۲ - در صورتی که برنامه اشکال داشته باشد، خطاها در پنجره Properties لیست می‌شوند.

۳ - با دستور namespace فضای نام جدید تعریف می‌شود

۴ - کلید میانبر برای ترجمه برنامه F6 است.

۵ - اگر محتوای پنجره Solution Explorer خالی باشد. یعنی برنامه اشکال تایپی دارد.

در سوالات زیر گزینه صحیح را انتخاب کنید.

۶-به صورت پیش‌فرض پروژه‌های ایجاد شده در برنامه ویژوال استودیو در کدام مسیر ذخیره می‌شود؟

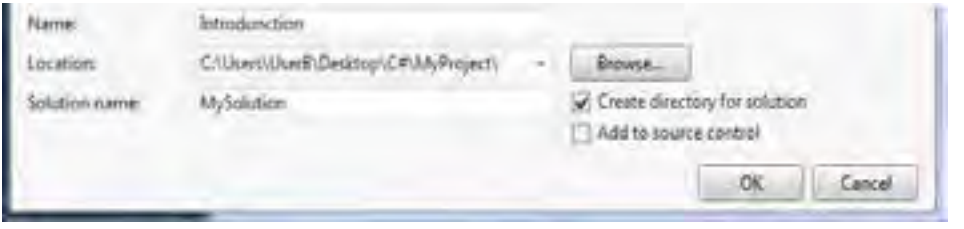
الف) My Documents \ Visual Studio \ Projects

ب) C:\ProgramFiles \ Windows \ System32

ج) My Documents \ Projects

د) C:\ Projects

۷- اگر هنگام تعیین نام برای یک پروژه جدید، در پنجرهٔ New Project اسامی زیر را وارد کنیم، کدام گزینه زیر ایجاد نخواهد شد؟



الف) پوشه Introdunction

ب) پوشه MyProject

ج) فایل MySolution.v11.suo

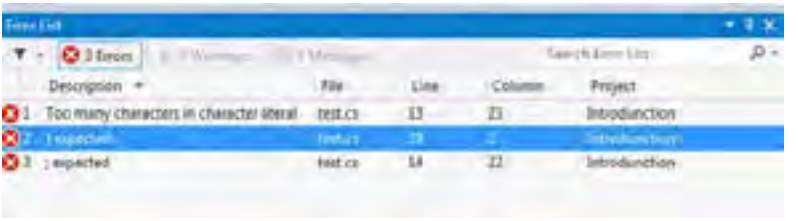
د) فایل MySolution.sln

۸-متن برنامه در کدام پنجره نگهداری می‌شود؟

الف) Tool Bar ب) Project Editor

ج) Error List د) Solution Explorer

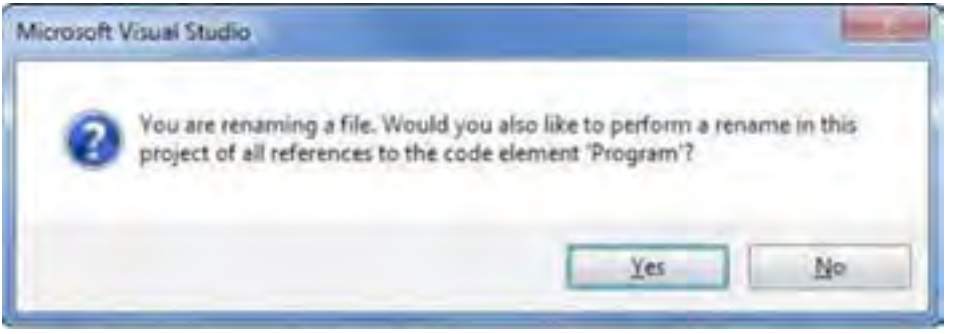
۹- در شکل زیر تعداد. .. . خطا در فایل. .. .. مشاهده می‌شود.



الف) یک – Test ب) سه – Test

ج) یک – Introduction د) سه - Introduction

۱۰- اگر در پنجره زیر کلید Yes را انتخاب کنیم، چه عملی انجام می‌شود؟



الف) تغییر نام پروژه در تمام مکان‌هایی که به این نام رجوع می‌شود.

ب)تغییر نام پروژه فقط در مکان جاری

ج)ترجمه پروژه به فایل اجرایی

د) ایجاد یک پروژه جدید در ویژوال استودیو

در جملات زیر، جاهای خالی را با عبارات مناسب پرکنید

۱۱-کلید F5 میانبری برای ..... است.

۱۲-در برنامه سی شارپ کل برنامه و بلاک‌ها در علامت ..... نوشته می‌شود.

۱۳-برای ایجاد صدا در برنامه از متد ..... استفاده می‌شود.

۱۴-برای معرفی یک فضای نام از دستور..... استفاده می‌شود.

۱۵-در محیط IDE کلمات کلیدی به رنگ ..... نشان داده می‌شوند.

ارتباط تمارین فصل و اهداف رفتاری

ردیف	اهداف رفتاری فصل سوم	خودآزمایی	تمرینات برنامه‌نویسی
۱.	IDE را تعریف کند و برای آن مثال بیاورد.	۱و۲	
۲.	مزایای استفاده از IDE و کاربرد VS را بیان نماید.	۳و۴و۵و۶و۷	
۳.	یک برنامه‌ی جدید در محیط VS از نوع کنسول ایجاد کند، آن را ذخیره، ترجمه و اجرا کند .		۱ تا ۴

بودجه بندی فصل ها

فصل سوم
هفته دوم بهمن ماه

موضوع : آشنایی با ویژوال استودیو

هدف کلی: آشنایی با نصب ویژوال استودیو و مفهوم IDE و نحوه کار با آن - شناخت پنجره های ویژوال استودیو

ساعات تئوری: ۲ **ساعات عملی:** ۶

فعالیتهای آموزشی موازی
-بررسی و حل تمرینهای فصل دوم
-مرور واژگان
- آموزش استفاده از Msdn (help)

فعالیتهای رقابتی و ارزشیابی
- ارزشیابی تشخیصی فصل دوم
- آزمون عملی دو به دوی دانش آموزان از یکدیگر

فصل ۴

آشنایی با انواع داده ها و متغیرها

اهداف رفتاری

متغیر را تعریف کند و انواع متغیر را در برنامه های خود به کار بندد.

انواع داده ها را نام ببرد و تفاوت کاربرد هر یک را توضیح دهد.

میزان حافظه و محدوده انواع داده ها را بیان کند.

متغیرها را به طور صحیح در برنامه اعلان کند و آن ها را مقداردهی نماید.

شکل نمایش تقطه شناور را توضیح دهد و اعداد اعشاری را در این قالب بنویسد.

از متد()ReadLine برای دریافت داده های یک برنامه از ورودی استفاده کند.

بر روی رشته دریافتی از ورودی، تغییراتی داده و سپس نمایش دهد.

از متد ()Parse برای تبدیل یک رشته به یک عدد استفاده کند.

آشنایی با انواع داده ها و متغیرها

کلیدواژه ها : نوع داده ، متغیر، Write ، WriteLine، ReadLine ، Read ، ReadKey

مقدمه

همان طور که می‌دانید متغیر[1] محلی از حافظه با یک نام نمادین می‌باشد. آنچه در کد به آن ارجاع می‌شود نام متغیر است اما کامپایلر این نام را با مکان حقیقی حافظه جایگزین می‌کند. در تعریف یک متغیر، حوزه تعریف و نوع داده‌ای آن اهمیت ویژه‌ای دارند. حوزه تعریف مشخص می‌کند که متغیر در چه محدوده ای قابل شناسایی باشد[2]. به عنوان مثال اگر یک متغیر به صورت محلی و در بلوک خاصی تعریف گردد تنها در همان محدوده قابل شناسایی و استفاده است و اگر به صورت public تعریف گردد در کل پروژه و بیرون از آن قابل دستیابی است.

مثال:

```
class test
  {
    public int x;
    private void function()
    {
      int y;
    }
    }
```

در این قطعه کد، x یک متغیر عمومی (public) و y یک متغیر محلی (local) است.

۴-۱- نوع داده ای

تعیین نوع داده‌ای در زبان C# ضروری می‌باشد. هرنوع داده، مجموعه‌ای از مقادیر به همراه مجموعه ای از عملیات را مشخص می‌کند.

---

۱ variable

۲ از جمله محدوده ها می‌توان از public،private،local، protected و internal نام برد.

جدول شماره ۴-۱ انواع داده‌ای را در زبان#C نشان می‌دهد. باید توجه داشت که تمام انواع داده‌ای ذکر شده، به استثناء انواع string و object (که از نوع class می‌باشند)، خود از نوع struct هستند.

۴-۱-۱- انواع داده ها در #C

- لازم است بدانید اسامی که برای معرفی انواع داده در ویرایشگر #C به کار می‌رود، به انواع داده‌های FCL[۳] نگاشت شده است تا برنامه نویسان راحت تر با انواع داده‌ها کار کنند. به جدول زیر دقت کنید.

۳ Framework Class Library

محدوده	پسوند	تعداد بایت	تعداد بیت	نوع داده	نوع داده NET.	نوع داده C#
0 to 255		1	8	Unsigned integer	Byte	byte
-128 to 127		1	8	Signed integer	SByte	sbyte
-32,768 to 32,767		2	16	Signed integer	Int32	short
0 to 65535		2	16	Unsigned integer	UInt16	ushort
-2,147,483,648 to 2,147,483,647		4	32	Signed integer	Int32	int
0 to 4294967295	U	4	32	Unsigned integer	UInt32	uint
-9223372036854775808 to 9223372036854775807	L	8	64	Signed integer	Int64	long
0 to 18446744073709551615	UL	8	64	Unsigned integer	UInt64	ulong
-3.402823e38 to 3.402823e38	F	4	32	Single-precision floating point type	Single	float
-1.79769313486232e308 to 1.79769313486232e308	D	8	64	Double-precision floating point type	Double	double
Unicode symbols used in text U+0000 to U+ffff		2	16	A single Unicode character	Char	char
True or false		1	8	Logical Boolean type	Boolean	bool
				Base type of all other types	Object	object
				A sequence of characters	String	string
±1.0 × 10e−28 to ±7.9 × 10e28	M	16	128	Precise fractional or integral type*	Decimal	decimal

Precise fractional or integral type that can represent decimal numbers with 29 significant digi

تعریف متغیر مانند دیگر زبان‌های شبه C به صورت زیر است:

نوع داده‌ای نام متغیر ;

```
int  x ;
```

و برای تعریف هم زمان چند متغیر از یک نوع نیز می‌توان به گونه زیر عمل کرد:

نوع داده‌ای نام متغیر اول , نام متغیر دوم , ... ;

```
int  x , y , z ;
```

- از کلمه کلیدی var نیز می‌توان برای تعریف متغیر استفاده نمود. بدین ترتیب کامپایلر در هنگام کامپایل با توجه به مقادیری که می‌تواند در متغیر قرار بگیرد یک نوع داده‌ای را برای آن متغیر بر می‌گزیند. در واقع نوع متغیر به صورت ضمنی تعیین می‌شود و حتما باید در هنگام معرفی مقدار دهی اولیه شود.

```
var x = 1000;

Console.WriteLine(x.GetType());
```

از طریق متد GetType() می‌توانیم نوع داده ای تعیین شده برای متغیر را دریابیم.

نوع داده ای x در مثال بالا، System.Int32 است.

- کد نوع char طبق استاندارد یونیکد (Unicode) است. دراستاندارد یونیکد، کد هر کاراکتر عددی بین ۰ تا۶۵۵۳۵ (۱۶ بیت) است و تمام نشانه‌ها، علائم و حروف الفباء زبانهای کشورهای مختلف به وسیله این استاندارد کدبندی شده است. این کدبندی مستقل از سیستم عامل، زبان برنامه نویسی و سخت افزار است[۱].

با توجه به این که Unicode دو بایتی است و نمایش کدی که برای کاراکترها در نظر گرفته شده عدد بزرگی می‌شود بنابر این برای نمایش کد بجای مبنای ۲ از مبنای ۱۶ استفاده می‌شود که برای هر ۴ بیت یک رقم مبنای ۱۶ در نظر گرفته می‌شود. مثلا کد ‘A’ که ۶۵ است به شکل ‘\u0041’ یا ‘\x0041’ یا ‘\u41’ یا ‘\x41’ نمایش داده می‌شود .

۱ می‌توانید کد کاراکترهای مختلف را در سایت http://unicode-table.com/en مشاهده نمایید.

۴-۲- نمایش عبارات و محتوای متغیرها [۱]

برای نمایش نتایج پردازش و اطلاعات می‌توان ازمتد Write یا WriteLine استفاده نمود. در اینجا چند روش کاربرد این دو متد را مورد بررسی قرار می‌دهیم :

روش اول

در این روش از یک **عبارت رشته‌ای** در دستور WriteLine، برای نمایش خروجی مورد نظر و مقادیر متغیرها استفاده می‌شود و برای اتصال عبارت‌ها، بین رشته‌ها و متغیرها از علامت + استفاده می‌شود.

به مثال زیر توجه کنید :

```
int a =10;
float   b=12.53781f;
System.Console.WriteLine( " output :  " + a + "   " + b );
```

خروجی :

```
output:  10    12.53781
```

۱ برای مطالعه بیشتر نحوه نمایش مقادیر می توان به آدرس زیر مراجعه نمود :
http://msdn.microsoft.com/en-us/library/system.console.writeline(v=vs.110).aspx

روش دوم

در روش قالب بندی با استفاده از آکولاد باز و بسته {} و شماره داخل آن محتوای متغیر و یا محاسبات مورد نظر نمایش داده می‌شود، به این نکته توجه کنید که شماره داخل آکولادها از صفر شروع می‌شود. عدد صفر در رشته با قالب بندی خاص نشان دهنده اولین آرگومان، عدد ۱ نشان دهنده دومین آرگومان و ... می‌باشد.

مثال:

```
Console.WriteLine ( " {0} , {1} , {2} " ," One", 5 ," seven  ");
```

خروجی :

```
One , 5 , seven
```

در مثال بالا رشته با قالب خاص "{2} , {1} , {0}" آرگومان اول {0} با رشته "One" و {1} با عدد ۵ و {2} با رشته "seven" جایگزین می‌شود. چون در عبارت توضیحی بین {} ها علامت کاما (,) قرار دارد در خروجی نیز این علامت نمایش داده می‌شود. پس هر عبارتی که در عبارت توضیحی نوشته شود، عینا در خروجی نمایش داده خواهد شد.

- **در** قالب خاص می‌توان یک آرگومان را بیشتر از یک بار در خروجی چاپ کرد.

مثال:

```
Console.WriteLine("{0} {0} {1} {0} {1}", "hello", "world ");
```

خروجی :

```
hello hello world  hello world
```

- از آنجایی که علائم آکولاد باز و بسته برای نشان دادن شروع و پایان رشته قالب بندی به کار می‌روند پس چگونه این علائم را می‌توان در خروجی نمایش داد؟

```
Console.WriteLine("{{ {0} }}", 7);
```

خروجی :

```
{ 7 }
```

برای نمایش آکولاد کافیست آکولاد باز یا بسته را دو بار تکرار کنید تا یک بار نمایش داده شود. به کد بالا توجه کنید، مشاهده می‌کنید که عدد صفر به اولین آرگومان بعد از رشته با قالب بندی خاص یعنی عدد ۷ اشاره می‌کند و در نتیجه عدد ۷ همراه با آکولاد باز و بسته در اطراف آن در خروجی چاپ می‌شود.

روش سوم

در قالب بندی اعداد، می‌توان اعداد را با فرمت‌های خاصی و مورد نیاز نمایش داد.

مثال:

```
Console.WriteLine("{0:C}", 500);
```

خروجی :

```
$500.00
```

همان طور که می‌بینید می‌توان عدد صحیح ۵۰۰ را با استفاده از فرمت **C**: به حالت پولی در آورد. برای این کار می‌توان از فرمت "{0:C}" استفاده کرد. برای اعشاری کردن عدد ۵۰۰ می توان از فرمت "{0:F3}" به صورت زیر استفاده کرد.

مثال:

```
Console.WriteLine ("{0:F3}", 500);
```

خروجی :

```
500.000
```

همـان طـور کـه می‌بینیـد فرمـت "{0:F3}" سـه رقـم اعشـار بـه عـدد ۵۰۰ اضافـه می‌کنـد. بـرای اعمـال دو رقـم اعشـار کافیسـت عـدد بعـد از F را بـه ۲ تغییـر دهیـد .

در جـدول شـماره ۴-۲ لیسـتی از توصیـف کننـده هـای فرمـت[۱] بـرای قالـب بنـدی اعـداد آمـده اسـت :

جدول۴-۲ : فرمت های قالب بندی اعداد

Format	Specifier
مقدار پولی خاص یک محل	C
انواع صحیح(integer)	D
نماد علمی	E
نقطه اعشار ثابت	F
اعداد عمومی	G
نقطه اعشار ثابت با جدا کننده کاما	N
اعداد دارای درصد	P
هگزادسیمال	X

۱ format specifiers

نکته

کوچک یا بزرگی حروف Specifier تفاوتی نمی‌کند.

تاثیر دقت یک قالب خاص بستگی به تنظیمات منطقه ای دارد. به عنوان مثال قالب پولی C به طور خودکار قالب پولی منطقه انتخاب شده را نشان می‌دهد. برای اکثر کاربران این اطلاعات به طور پیش فرض مربوط به منطقه و زبان آنها است.

در جدول شماره ۴-۳ چندین قالب عددی نشان داده شده است. به تاثیر اعشار در قالب های مختلف توجه کنید.

جدول۴-۳ : فرمت های قالب بندی اعداد اعشاری

متغیر	قالب	خروجی
double v = 17688.65849;	“{0:F2}”	17688.66
double v = 17688.65849;	“{0:N5}”	17, 688.65849
double v = 17688.65849;	“{0:e}”	1.768866e+004
double v = 17688.65849;	“{0:r}”	17688.65849
double v = 0.15;	“{0:p}”	15.00 %
int x = 21;	“{0:X}”	15
int x = 21;	“{0:D12}”	000000000021

با استفاده از متد ToString() می‌توان عدد را به معادل رشته ای آن با قالب بندی خاص تبدیل کرد.

مثال:

```
int x = 500;

Console.WriteLine(x.ToString("C")); // $500.00
```

روش چهارم

با استفاده از قالب های مشخصی می‌توان حداقل طول رشته ها را معین کرد.

مثال:

```
Console.WriteLine("{0,10}", "hello");

Console.WriteLine("{0,-10}{1}", "hello", "world");
```

خروجی :

```
     hello
hello     world
```

همان طور که مشاهده می‌کنید در مثال بالا از قالب {x,y} استفاده شده است که در آن x آرگومان عددی و به معنای رشته مورد نظر و y حداقل طول آن است. در خط اول قالب {0,10} برای رشته hello ، ۱۰ مکان در نظر می‌گیرد ، پس ۵ فاصله قبل از hello خالی می‌ماند. در خط دوم کد بالا قالب {0,-10} مانند قالب {0,10} رشته hello را در محدوده ۱۰ تایی چاپ می‌کند ولی چون علامت عدد منفی است فضای خالی در سمت راست رشته قرار می‌گیرد. در زمانی که نیاز به فاصله گذاری باشد، طول مثبت عدد را راست چین کرده و فضای خالی به سمت چپ آن اضافه شده و طول منفی عدد را چپ چین کرده و فضای خالی به سمت راست رشته اضافه می‌شود. مقدار فضای خالی به طول رشته بستگی دارد. به عنوان مثال اگر طول یک رشته ۵ باشد

- اگر حداقل طول رشته معین شده در دستور از طول رشته چاپی کمتر باشد، فاصله‌ای ایجاد نمی‌شود و رشته کامل چاپ خواهد شد.

قالب بندی عددی می‌تواند با قالب بندی حداقل طول رشته نیز ترکیب شود :

```
Console.WriteLine("{0,10:C}", 500);
```

خروجی :

```
$500.00
```

در مثال بالا، ۱۰ فضای خالی برای قالب بندی تعیین شده است و در نتیجه چون طول عدد، سه است، باید ۷ فاصله در سمت چپ آن اضافه شود ولی چون علامت $ هم یک فضا را اشغال کرده است و همچنین ۲ عدد اعشار هم به عدد اضافه می کند به جای ۷ فاصله با احتساب علامت ممیز، ۳ فاصله خالی در سمت چپ عدد ۵۰۰ باقی می‌ماند.

روش پنجم

سی شارپ اجازه قالب بندی اعداد به صورت سفارشی را نیز می‌دهد. برای این کار میتوان در دستور Write یا WriteLine کاراکترهای خاصی را بکار برد.

جدول۴-۴ : فرمت های قالب بندی اعداد به صورت سفارشی

کاراکتر	معنی
#	عدد*
.	اعشار
,	جدا کننده سه رقمی
%	درصد
0	عدد *
;	اعداد مثبت ، منفی و مقادیر صفر را با قالب های خاصی نمایش می دهد.
E0 E+0 E-0 e0 e+0 e-0	نماد علمی

* تفاوت # و 0 در این مورد است که 0 مشخص کننده یک عدد قطعی می‌باشد و اگر در محل آن عددی وجود نداشته باشد 0 جایگزین می‌شود. اما در محل # اگر عددی وجود نداشته باشد جایگزینی لحاظ نمی‌شود.

مثال:

```
Console.WriteLine("{0:##}", 7);
```

خروجی:

07

7

هم چنان که مشاهده می‌کنید در خط اول که با قالب بندی 00 مشخص شده است در محل 0 دوم ۷ قرار می‌گیرد و در محل 0 اول چون رقمی موجود نیست خود عدد قرار می‌گیرد اما در خط دوم بعد از جایگزینی # دوم با ۷ در محل # اول رقمی جایگزین نمی‌شود. توجه کنید که:

۱. علامت نقطه (.) مکان اعشار را مشخص می‌کند.

۲. علامت # می‌تواند در برگیرنده عددی باشد که قرار است در سمت چپ یا راست آن اعشار قرار بگیرد. اگر این علامت در سمت راست اعشار قرار بگیرد دقت اعشار را مشخص می‌کند و ممکن است در صورت لزوم عدد را گرد کند و اگر در سمت چپ ممیز باشد نماینده قسمت صحیح عدد می‌باشد. اگر طول عدد از تعداد علامتهای # که در سمت چپ علامت اعشار قرار دارند بیشتر باشد، کل عدد نشان داده می‌شود. به عنوان مثال اگر عدد مورد نظر شما ۱۲۳۴۵ باشد و شما از سه علامت ### قبل از ممیز استفاده کرده باشید همه عدد نشان داده می‌شود. اگر مقدار اصلی دارای اعشار باشد و شما در قالب بندی قسمت اعشار را قالب بندی نکنید، عدد اصلی گرد می‌شود. به عنوان مثال اگر مقدار اصلی ۱۲۳/۴۵ باشد و شما از ### استفاده کنید عدد ۱۲۳ نمایش داده می‌شود.

```
Console.WriteLine("{0:00##.#00}", 21.3); // 0021.300
```

۳. برای اعداد بزرگ هم می‌توانید برای جدا کردن ارقام، از علامت کاما استفاده کنید.

```
Console.WriteLine("{0:#,###.#}", 3421.3); // 3,421.3
```

لازم نیست که مکان کاما را برای همه قسمت ها مشخص کنید چون خودش به صورت خودکار سه رقم سه رقم جدا می کند مثلا :

```
Console.WriteLine("{0:#,###.#}", 8763421.3); // 8,763,421.3
```

کاما معنای دیگری هم دارد. وقتی که قبل از علامت اعشار قرار می‌گیرد.

به عنوان یک مقیاس عمل می‌کند. هر کاما مقدار را بر ۱۰۰۰ تقسیم می‌کند.

```
Console.WriteLine("{0:#,###,.#}", 8763421.3); // 8,763.4
```

همان طور که در خروجی مشاهده می‌کنید مقیاس در اینجا به معنی هزار است.

می‌توان از کاراکتر های کنترلی \t یا \n در صورت لزوم استفاده کرد. کاراکتر \t برای ایجاد یک پرش (tab)و کاراکتر \n برای رفتن به خط جدید در خروجی به کار می‌رود.

۴. کاراکتر E و e برای نمایش اعداد به صورت نماد علمی به کار می‌روند. حداقل یک صفر و در صورت لزوم تعداد بیشتری صفر بعد از E و e می‌آیند. صفرها تعداد ارقام اعشار را نشان می‌دهند. استفاده از حرف E و یا e ممکن است باعث چاپ این حروف در خروجی شوند. در نتیجه برای تعریف توان مثبت و منفی می توان از علامت – و + بعد از این دو کاراکتر به صورت E+، e+ و یا E-، e- استفاده کرد.

۵. علامت ";" هم شما را قادر می‌سازد که اعداد مثبت، منفی و مقادیر صفر را با قالب های خاصی جدا سازید. شکل کلی سفارشی سازی با این کاراکتر به صورت زیر است:

```
Console.WriteLine("{0:#.##;(#.##);0.00}", num);
```

در مثال بالا اگر num مثبت باشد مقدار به صورت #.## (با دو رقم اعشار)، در غیر این‌صورت اگر منفی باشد مقدار به صورت (#.##) و اگر صفر باشد به صورت 0.00 نمایش داده می‌شود.

جدول ۴-۵ مثال کاربرد کاراکترهای ذکر شده را نشان می‌دهد.

جدول۴-۵ : کاربرد کاراکترهای فرمت قالب بندی اعداد اعشاری

متغیر	قالب	خروجی
double num = 64354.2345	“{0:#.##}"	64354.23
double num = 64354.2345	“{0:#,###.##}"	64,354.23
double num = 64354.2345	“{0:#,###.#}"	64,354.2
double num = 64354.2345	“{0:#.###e+00}"	6.435e+04
double num = 64354.2345	“{0:#%}"	6435423%
double num = 64354.2345	“{0:#0,}"	64
double num = 64354.2345	“{0:#.#;(#.##);0.00}"	64354.2
double num =- 64354.2345	“{0:#.#;(#.##);0.00}"	(64354.23)
double num = 0	“{0:#.#;(#.##);0.00}"	0.00

روش ششم

برای نمایش اعداد اعشاری بصورت استاندارد معمولا از روش نماد علمی استفاده می‌شود.

مانتیس ۱۰ * به توان نما

در تبدیل یک عدد اعشاری به نماد علمی باید توجه داشت که در مانتیس باید تعداد ارقام صحیح قبل از نقطه ممیز اعشار فقط یک رقم باشد و بر اساس جابجایی ارقام دیگر توانی از ۱۰ در نظر گرفته می شود تا عدد تغییر نکند.

۱۲۷/۳۴۵ → ۱/۲۷۳۴۵ * ۱۰ ^۲

۰/۰۰۲۸۳۵ → ۲/۸۳۵ * ۱۰ ^-۳

می‌توان با استفاده از کارکتر e یا E که در روش قبل ذکر شد، مقدار اعشاری را به فرم نماد علمی در خروجی نمایش داد.

مثال:

```
Console.WriteLine(“Default format: “ + num);

Console.WriteLine(“Use scientific notation:  {0:#.###e+00}”, num);
```

خروجی :

```
Default format: 64354.2345

Use scientific notation: 6.435e+04
```

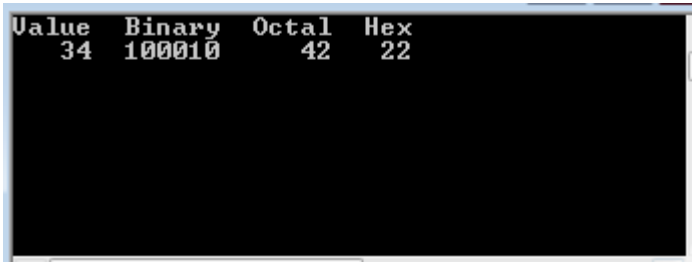
۴-۲-۱- اعداد صحیح در مبنای ۱۶ و نمایش آن

در زبان C# می‌توان اعداد صحیح را در مبنای ۱۶ نیز نوشت. برای این کار از پیشوند 0x یا 0X استفاده می‌شود. مثلا :

```
byte number = 0x12;  // number is 18 in decimal
```

برای تبدیل و نمایش اعداد در مبنای ۲ و ۸ و ۱۶ می‌توان از روش زیر استفاده کرد :

```
byte   number=34 ;
Console.WriteLine(“{0,5}{1,8}{2,7}{3,5}” , ”Value” , ”Binary” , ”Octal” , ”Hex”);
Console.WriteLine(“{0,5}{1,8}{2,7}{3,5}” ,number,
Convert.ToString(number, 2),
Convert.ToString(number, 8),
Convert.ToString(number, 16));
```



در ایـن مثـال تابـع ToString ، یـک عـدد صحیـح و عـدد مبنـای مـورد نظـر را از ورودی می‌گیـرد و عـدد را بـه مبنـای مـورد نظـر تبدیـل می‌کنـد و حاصـل را بـه شـکل رشـته برمی‌گردانـد.
در دستور writeLine عبارت {0} که قبلا توضیح داده شد، عبارت اول را چاپ می‌کند.
عبـارت {0,5} بـرای عبـارت اول محـدوده ای بـه طـول ۵ در صفحـه خروجـی در نظـر می‌گیـرد ایـن روش بـرای زیبـا سـازی و بـالا بـردن خوانایـی خروجـی اسـتفاده می‌شـود.

۴-۳- دریافت رشته و مقادیر ورودی

بـرای دریافـت مقادیـر ورودی می‌تـوان از متـد هـای زیـر اسـتفاده نمـود. ایـن متدهـا درکلاس Console تعریـف شـده‌اند و در فضـای نامـی System قـرار دارنـد.

روش اول

متـد Read یـک کاراکتـر از ورودی می‌گیـرد و کـد آن را برمی‌گردانـد. همانطـور کـه در تعریـف ایـن متـد مشـخص اسـت خروجـی ایـن متد int اسـت.

```
public static int Read()
```

این متد به صورت زیر استفاده می شود:

```
int x;
x=Console.Read();
```

کد اسکی کاراکتر خوانده شده در متغیر x مقداردهی می‌شود.

روش دوم

متد ReadKey یک کاراکتر می‌گیرد و اطلاعات آن را تحت ساختار ConsoleKeyInfo برمی‌گرداند و نیازی به زدن enter نیست.

```
public static ConsoleKeyInfo ReadKey()
```

این متد به صورت زیر استفاده می‌شود:

```
ConsoleKeyInfo a = Console.ReadKey();
```

در نتیجه اطلاعات کاراکتر خوانده شده، در متغیر a مقداردهی می‌شود.

- اگر یک حرف وارد شود و بخواهیم با استفاده از این متد حرف وارد شده را بررسی کنیم کافیست که از ویژگی KeyChar استفاده نماییم.

```
char c=Console.ReadKey().KeyChar;
```

روش سوم

متدReadLine با زدن enter ، یک رشته از ورودی می‌گیرد. همانطور که در تعریف این متد مشخص است خروجی این متد string است.

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

این متد به صورت زیر استفاده می‌شود:

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

رشته خوانده شده در متغیر s مقداردهی می‌شود.

این متدها را به صورت زیر هم می توان استفاده نمود:

```
System.Console.Read();
System.Console.ReadKey();
System.Console.ReadLine();
```

کامپیوتر با اجرای این متدها متوقف شده و منتظر دریافت داده می‌شود.

۴-۳-۱- نحوه دریافت مقادیرغیر رشته ای از ورودی با دستور ReadLine

دستور( )Console.ReadLine منتظر می‌ماند تا کاربر ورودی وارد کرده و سپس کلید Enter را فشار دهد. این دستور عبارت ورودی را به فرم رشته دریافت می‌نماید.

اکنون اگر کاربر مقدار عددی وارد کند، ورودی کاربر به فرم رشته‌ای دریافت می‌شود و در یک متغیر از نوع رشته‌ای قرار می‌گیرد. اما می دانیم که برای اجرای محاسبات و یا تصمیم‌گیری بر روی اعداد نمی‌توان از آنها در فرم رشته‌ای استفاده کرد و باید آنها را بصورت عددی مورد استفاده قرار داد. به همین منظور باید متغیر رشته ای را به نحوی به مقدار عددی تبدیل نماییم.

برای این منظور سه روش وجود دارد :

الف – استفاده از متد Parse

ب – استفاده از متد Convert

ج – استفاده از متد TryParse

ایـن متدهـا یـک رشـته را دریافـت کـرده و مقـداری از نـوع فراخوانـنـده اش بـاز می‌گردانـد. بـا مثـال ایـن متـد هـا را بـا هـم مقایسـه می‌کنیـم.

۴-۳-۱-۱ – استفاده از متد Parse

- متد int.Parse(string s) ، رشته s را به معادل عددی آن تبدیل می‌کند و در متغیر عددی مقداردهی می‌کند.

بررسی حالات مختلف:

```
int result;
```

اگر مقدار ورودی، رشته عددی باشد ، به عدد تبدیل می‌شود.

```
result = int.Parse(“1234”); //-- 1234
```

اگر مقدار ورودی رشته غیر عددی باشد، پیغام خطای FormatException را نمایش می‌دهد.

```
result = int.Parse(“1234.65”); //-- FormatException
```

اگر مقدار ورودی رشته null باشد، پیغام خطای ArgumentNullException را نمایش می‌دهد.

```
result = int.Parse(null); //-- ArgumentNullException
```

اگر مقدار ورودی، رشته عددی بزرگتر از محدوده int باشد، پیغام خطای OverflowException رانمایش می دهد.

```
result = int.Parse(“98765439873”); //-- OverflowException
```

۴-۳-۱-۲ – استفاده از متد Convert

متد Convert.ToInt32 (string s)، رشته s را به معادل عددی آن تبدیل می‌کند و در متغیر عددی مقداردهی می‌کند:

بررسی حالات مختلف:

```
int result;
```

- اگر مقدار ورودی رشته عددی باشد، به عدد تبدیل می شود.

```
result = Convert.ToInt32("1234"); //-- 1234
```

- اگر مقدار ورودی رشته غیر عددی باشد، نتیجه پیغام خطای FormatException را نمایش می دهد.

```
result = Convert.ToInt32("1234.65"); //-- FormatException
```

- اگر مقدار ورودی رشته null باشد، مقدار ۰ را برمی‌گرداند.

```
result = Convert.ToInt32(null); //-- 0
```

- اگر مقدار ورودی رشته عددی بزرگتر از محدوده int باشد، نتیجه پیغام خطای OverflowException را نمایش می دهد.

```
result = Convert.ToInt32("98765439873"); //-- OverflowException
```

۴-۳-۱-۳ – استفاده از متد TryParse

متد int.TryParse(string s , out int result) رشته s را به معادل عددی آن تبدیل می‌کند و در آرگومان دوم در متغیر جلوی عبارت out مقداردهی می‌کند و بر اساس نحوه عملکرد و درستی یا نادرستی تبدیل، true یا false برمی گرداند.

بررسی حالات مختلف:

```
int result;
bool success;
```

- اگر مقدار ورودی رشته عددی باشد، به عدد تبدیل می شود و مقدار true را برمی‌گرداند که نشانه دهنده این است که تبدیل با موفقیت انجام شده است.

```
success = int.TryParse("1234", out result);//--success =>true; result => 1234
```

- اگر مقدار ورودی، رشته غیر عددی باشد، مقدار ۰ را برمی‌گرداند و درستی عملکرد تبدیل را false برمیگرداند.

```
success = int.TryParse("1234.65", out result);//--success => false; result => 0
```

- اگر مقدار ورودی، رشته null باشد، مقدار ۰ را برمی گرداند و درستی عملکرد تبدیل را false برمی گرداند.

```
success = int.TryParse(null, out result);//--success => false; result => 0
```

- اگر مقدار ورودی ، رشته عددی بزرگتر از محدوده int باشد ، مقدار ۰ را برمیگرداند و درستی عملکرد تبدیل را false برمی‌گرداند.

```
success = int.TryParse("98765439873", out result);//--success => false; result
=> 0
```

TryParse مانند Convert عمل می‌کند. مزیت TryParse این است که در صورتی که متد تبدیل با خطا مواجه شود، برنامه می تواند آن را مدیریت کند.

۴-۴- مقایسه ای با سایر زبان ها

انواع داده‌ای در زبان های مختلف می‌توانند از جهت تنوع انواع، جنس و رفتار آنها در هنگام مقدار دهی تفاوت‌هایی داشته باشند. به عنوان مثال در بعضی از زبان ها مانند C و C++ اگر مقداری خارج از ظرفیت متغیر در آن مقداردهی کنیم، برنامه خطای زمان اجرا یا کامپایل نمی‌گیرد و پدیده over flow رخ می‌دهد. بدین معنی که عدد دیگری در متغیر ذخیره می‌شود.

مثلا در زبان C و C++ محدوده مجاز متغیری از نوع byte بین ۰ تا ۲۵۵ است. حال اگر عدد ۲۵۶ در متغیری از نوع byte مقداردهی شود، مقدار ۰ در آن متغیر مقداردهی می‌شود.

مزیت گرفتن خطای محدوده غیر مجاز در C# این است که بر ورود داده ها و محدوده آن ها، کنترل دارد و عیب آن این است که در صورت اینکه بنا به دلایلی مقدار متغیری در زمان اجرا خارج از محدوده شود، برنامه خطای زمان اجرا می‌گیرد و برنامه نویس باید برای اینکه برنامه قطع نشود، از دستورات مناسب استفاده نماید و یا خطا را کنترل نما ید .

همچنین دستورات خروجی و ورودی در زبان های مختلف حتی با یکسان بودن محیط اجرا، متفاوت خواهند بود. برای نمونه مروری اجمالی می کنیم بر دستورات ورودی و خروجی در سه زبان C،C++ و C# که دارای محیط اجرای مشابهی هستند.

جدول۴-۶ : جدول تطبیقی دستورات ورودی و خروجی با زبان های دیگر

نوع دستور	C	C++	C#
دستور ورودی	scanf	cin	Read , ReadLine
دستور خروجی	printf	cout	Write , WriteLine

واژگان

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	**Convert**	**تبدیل کردن**
۲	**Data type**	**نوع داده**
۳	**Format specifier**	توصیف کننده فرمت
٤	**Variable**	**متغیر**

پرسش و پاسخ

۱- اگر متغیری تعریف شود و مقداردهی نشود ، چرا کامپایلر #C خطا می گیرد؟

کامپایلر پیغام Warning می دهد که این متغیر استفاده نشده است. ولی برنامه قابل کامپایل و اجرا است.

۲- چگونه می توانیم محتوای یک متغیر را برابر Null قرار دهیم؟ کاربرد آن را ذکر کنید.

با استفاده از نوع داده Nullable که با علامت سوال بعد از تعیین نوع متغیر می‌آید مثلا int? a = null;

کاربرد آن در بهبود کار با پایگاه داده ها می‌باشد.

۳- تفاوت متد Read با ReadLine چیست؟

متد Read برای دریافت یک کاراکتر از ورودی به کار می رود (در صورتی که چند کاراکتر وارد شود فقط اولین کاراکتر را به عنوان ورودی در نظر می گیرد) و کد اسکی کاراکتر دریافتی (عدد صحیح) را برمی گرداند یعنی خروجی ان از نوع char است.

متد ReadLine برای دریافت یک رشته از ورودی به کار می رود و رشته ای که تا قبل از فشار دکمه Enter وارد شده است را برمی گرداند.

۴- معنای استاتیک (static) برای یک متد چیست؟

یعنی بدون ساخت شیء می توان مستقیماً از داخل کلاس، آن را فراخوانی کرد. یک مشخصه یا متد استاتیک متعلق به خود type است. یعنی اگر از آن object نسازید و instance نگیرید مستقیماً قابل فراخوانی است.

۵- چرا فقط در کنسول باید متد Main را از نوع static تعریف کرد و اگر static استفاده نشود خطا می گیرد؟

زیرا C# در کنسول، شیء از نوع کلاس اولیه نمی سازد و برای اجرای مستقیم تابع در کلاس بدون ساخت instance، باید استاتیک باشد.

۶- اگر ثابت عددی صحیح، بدون هیچ نمادی در انتهای آن استفاده شود، C# آن را از چه نوعی در نظر می‌گیرد؟

C# آن را از نوع int در نظر می گیرد.

واحد کار

واحد کار شماره: ۴-۱	فصل۴ - صفحه۵۹ تا ۷۲	مدت آموزش: ۱۳۵ دقیقه	روش تدریس پیشنهادی: پرسش و پاسخ.
عنوان: کاربرد متغیر		هدف کلی: آموزش متغیر و انواع آن	
اهداف رفتاری: انتظار می رود دانش آموز ۱. کاربرد و مفهوم متغیر را شرح دهد. ۲. انواع داده ها را نام ببرد و میزان حافظه، محدوده و تفاوت کاربرد هر یک را توضیح دهد. ۳. متغیرها را به طور صحیح اعلان کند و آن ها را مقداردهی نماید.		اهداف پنهان درس: آشنایی با برخی خطاهای مربوط به مقداردهی متغیرها هدف ایده آل :	
فرایند تدریس: از دانش آموزان می خواهیم چند عدد از ظروف آشپزخانه را نام ببرند. کارایی هر کدام را از دانش آموزان سوال می کنیم. با توجه به مراحل قبل، با طرح پرسش های زیر ذهن دانش آموزان را متوجه نکات مهم می کنیم: آیا امکان دارد تمام سوپ یک قابلمه را در یک لیوان بریزیم؟(ظرفیت و حجم) آیا امکان دارد که آب درون یک پارچ در درون یک بشقاب جا شود؟ (ظرفیت).آیا زمانی که بشقاب غذا خالی شد می توان غذای دیگری دوباره در آن ریخت؟(پر کردن داده با داده های مختلف). آیا می توان از بشقاب تخت(پلوخوری)، بجای کاسه استفاده نمود؟ (نوع داده). آیا محتویات یک بشقاب برنج در درون یک قابلمه خالی جا می شود؟(ظرفیت یک متغیر و استفاده یک نوع داده کوچکتر در نوع داده بزرگتر و همجنس). آیا می توان نام ظروف یا کارایی آنها را تغییر بدهیم ؟ از دانش آموزان می خواهیم: ۱.نام ظروف و کارایی هر یک را، در خطوط جداگانه یاداشت نمایند.۲.حجم هر ظرف را با دیگری مقایسه کنند. در انتها با استفاده از مثال، به بیان مفاهیم متغیر، انواع داده ی و نکات مربوط به آن می پردازیم و از دانش آموزان می خواهیم مثال هایی در این رابطه بزنند.			

واحد کار شماره : ۴-۱	فصل۴ - صفحه۵۹ تا ۷۲

توصیه های اجرایی:

۱. نوع داده را کاملا به دانش آموزان با مثال های متنوع برای دانش آموزان شرح دهید.
۲.در نامگذاری متغیر ها، انتخاب اسم مناسب را تاکید نمایید.
۳.چگونگی استفاده از نوع داده اعشاری معمولی را به دانش آموزان توضیح دهید.

ارزشیابی از انتظارات:

۱. برای ذخیره نمره دانش آموز متغیر ، از نوع داده ای bool تعریف می شود. ص غ
۲. برای ذخیره نمودن نام خود از نوع داده ای string استفاده می کنیم. ص غ
۳. نوع داده ای uint در حافظه ۲۰ بایت فضا اشغال می کند. ص غ
۴. دستور لازم برای ذخیره عدد ۲۵۹ را دریک متغیر نوع صحیح بنویسید..
۵. مقدار فضای اشغالی long در حافظه ..... بایت می باشد.
۶. برای ذخیره اعداد گنگ از نوع داده ای .....با دقت زیاد استفاده کرد.

تکالیف و پروژه های پیشنهادی:

۱- برای نگهداری هر یک از داده های زیر در برنامه، یک متغیر با نوع داده ای مناسب تعریف نمایید.
الف)ظرفیت یک فلش مموری ب)سن افراد ج)شماره دانش آموزی د) وضعیت رسمی یا غیر رسمی بودن یک کارمند

۲- پس از اجرای دستورات زیر پیغام خطایی از کامپایلر زبان صادر می شود. علت خطا را در دستورات یافته و آن را بنویسید.

```
A);float c=oxa0f
B)int g G=3*4*5;
```

۳- مقدار عددی 12+3.1214414E را می توان در متغیر float یا ..... ذخیره کرد.

نمونه سوال از فصل ۴

درستی یا نادرستی هر عبارت را تعیین کنید. د برای درست و ن برای نادرست

۱-در متغیر از نوع داده sbyte اعدادی ۰ تا ۲۵۵ را می توان قرار داد.

۲-داده string در زبان برنامه نویسی سی شارپ از نوع غیر عددی است

۳-در سی شارپ برای ذخیره عدد در مبنای ۱۶ از پیشوند0x یا 0X استفاده می شود.

۴-نماد u یا U نشان دهنده نوع داده ulong است.

۵-برای دریافت اعداد در سی شارپ باید عمل خواندن وسپس تبدیل صورت گیرد.

در سوالات زیر گزینه صحیح را انتخاب کنید.

۶- برای تعریف متغیر از نوع اعشاری از کدام دسته نوع داده می توان استفاده کرد؟

الف) int, long, byte ب) single, double ج) double, float د) byte, sbyte

۷- پیغام 1 Constant value '300' cannot be converted to a 'byte' چه زمانی ظاهر می-شود؟

الف) یک متغیر از نوع byte تعریف شده باشد.

ب) مقداری به متغیر byte نسبت داده باشیم که فراتر از ظرفیت آن باشد.

ج) یک متغیر از نوع sbyte تعریف شده باشد.

د) در یک متغیر از نوع صحیح مقدار اعشاری قرار داده باشیم.

۸-در دستورات زیر متغیرهای PI و Mark در مجموع، چه ظرفیتی از حافظه را اشغال می کنند؟

```
double  PI=3.141592653589793238;
float   Mark = 17.75f;
```

الف) ۸ ب) ۱۲ ج) ۱۶ د) ۳۲

۹-پیام خطای زیر مربوط به کدام دستور است؟

1 Literal of type double cannot be implicitly converted to type 'float'; use an 'F' suffix to create a literal of this type

الف) float f = 45.7; ب) int a = 34 / 5.5; ج) byte b = 688;

در جملات زیر، جاهای خالی را با عبارات مناسب پرکنید.

۱۰-مناسب ترین نوع داده برای ذخیره کردن نمره درس برنامه سازی نوع ..... است.

۱۱-برای نشان دادن محتوای متغیرها بر روی صفحه نمایش از متد ..... استفاده می شود.

۱۲- برای ذخیره کردن یک کاراکتر در یک متغیر از نوع داده ..... استفاده می شود.

۱۳- در زبان برنامه نویسی سی شارپ، برای دریافت مقدار از ورودی از متد ..... استفاده میشود.

۱۴- انواع داده های زیر را در دو دسته داده عددی صحیح و اعشاری دسته بندی کنید.

double, sbyte, uint,float,short,decimal, ulong

۱۵- در یک خط بنویسید دستور زیر چه عملی انجام می دهد؟

number=number+1;

۱۶-دستوری بنویسید که نام کاربری شامل حروف و اعداد را از ورودی دریافت کند.

۱۷- دستوری بنویسید که نمره یک دانش آموز را از ورودی دریافت نماید.

جدول مشخصات آزمون فصول ۱-۲-۳-۴

جدول این بخش به دلیل پیچیدگی در قالب docx از طریق وبگاه tvoccd.sch.ir در اختیار هنرآموزان عزیز قرار خواهد گرفت.

نمونه آزمون مبتنی بر جدول مشخصات

درستی یا نادرستی هر عبارت را تعیین کنید. **د** برای درست و **ن** برای نادرست. (هر کدام ۰,۲۵ نمره)

۱- در کامپیوتر اطلاعات به عنوان ورودی پردازش می شود و نتیجه آن داده است که به دستگاه های خروجی ارسال می شود.

۲- برای تبدیل زبان برنامه نویسی سطح بالا به زبان ماشین از کامپایلر استفاده می شود.

۳- با اجرای دستور csc welcome.cs در محیط کنسول فایل برنامه welcome.cs ترجمه و اجرا می شود.

۴- تنها راه ترجمه برنامه استفاده از کلید میانبر F۶ است.

۵- اگر در برنامه یک عدد اعشاری بدون نشانه بنویسید این عدد به عنوان عدد اعشاری بادقت زیاد درنظر گرفته می شود.

در سوالات زیر گزینه صحیح را انتخاب کنید. (هر کدام ۰,۲۵ نمره)

۶- چنانچه پس از ترجمه خطایی مشاهده نشود و عمل ترجمه موفقیت آمیز باشد، در مسیر برنامه فایل با کدام پسوند ایجاد می شود؟

الف) .bat ب) .exe ج) .txt د) .net

۷- اینکه در بین نمرات درس یک دانشآموز نمره کمتر از ۰ و بیشتر از ۲۰ نداشته باشیم، نشان دهنده کدام مورد از فرایند رسیدن داده به اطلاعات است؟

الف) درستی انجام محاسبات ب) روش انجام پردازش

ج) صحت داده ها د) درستی برنامه

۸- پنجره Solution Explorer چه کاربردی دارد؟

الف) لیست خطاهای احتمالی را نشان می دهد.

ب) ساختار پروژه و تمام فایلهای موجود در آن را نشان می دهد.

ج) متن برنامه را نشان می دهد.

د) منوها و ابزارها را نشان می دهد.

۹- کدام گزینه صحیح است؟

الف) برای استفاده از یک متغیر باید آن را اعلان کرد.

ب) بدون مقداردهی به متغیر می توان از آن استفاده کرد.

ج) می توان از یک متغیر بدون اعلان و فقط با مقداردهی به آن، استفاده کرد.

د)می توان در برنامه چند متغیر هم نام اعلان کرد به شرطی که نوع آنها متفاوت باشد.

۱۰- در کدام گزینه متد Parse می تواند تبدیل را بدون مشکل انجام دهد؟

الف) int result = int.Parse("1234.65");

ب) char result = char.Parse("1234.65");

ج) float result = float.Parse("1234.65");

د) byte result = byte.Parse("1234.65");

در جملات زیر، جاهای خالی را با عبارات مناسب پرکنید.

۱۱- برای اینکه در ابتدای دستورات، فضای نامی System را ننویسیم، ابتدای برنامه کلمه System را جلو دستور ..... می نویسیم.

۱۲- به محیطی که در آن میتوان تمام مراحل برنامه نویسی، ترجمه، اشکال یابی و اجرا را انجام داد به صورت مخفف ..... می گویند.

۱۳- ConsoleColor دارای تعداد ..... مقدار رنگ در c# است.

۱۴- نشانه m انتهای یک عدد نشان دهنده نوع داده..... است.

۱۵- هر یک از انواع داده سمت راست را با میزان حافظه آن در سمت چپ مرتبط کنیدو در جای خالی بنویسید.(یک گزینه در ستون پاسخ اضافی است)(۱ نمره)

ushort	(---)	الف) ۱
long	(---)	ب) ۲
float	(---)	ج) ۱۶
bool	(---)	د) ۸
		ه) ۴

Trace و خطایابی

۱۶- اشکالات قطعه کد زیر چیست؟(۰,۵ نمره)

```
class  Program
{
 static void main()
{
   System.Console.Writeline(“hello reverybody”)
}
}
```

۱۷- نتیجه اجرای هر قطعه کد زیر چیست؟(هر کدام ۰,۵ نمره)

الف)
```
System.Console.Write(“ my favorite color is :”);
System.Console.WriteLine();
System.Console.Write(“Green”);
```

ب)
```
System.Console.WriteLine(“ my favorite color is :”);
System.Console.Write(“Green”);
```

۱۸- اشکالات قطعه کد زیر چیست؟ (۰,۵ نمره)

```
float k= 37.25;

sbyte a,b=100;

 a=b*2;
```

۱۹- نتیجه اجرای قطعه کد زیر چیست؟ (۱ نمره)

```
byte a = 17, b = 12;
Console.WriteLine(“the sum of a and a is :”+ (a + b) );
b = 7;
Console.WriteLine(“the sum of a and a is :” + (a + b));
Console.WriteLine(“the sum of a and a is :” + a + b);
```

سوالات تشریحی

۲۰- چگونه بدون ترجمه برنامه می توان از خطاهای برنامه باخبر شد؟ (۰,۵ نمره)

۲۱- ویژوال استودیو هنگام گزارش خطا، برای تشخیص محل خطا چه اطلاعاتی نمایش می دهد؟(۰,۵ نمره)

۲۲- تفاوت متد ReadKey() و ReadLine() چیست؟(۰,۵ نمره)

۲۳- برای نگهداری هر یک از داده های زیر دستور اعلان متغیر را بنویسید. نام متغیر باید صحیح و با مسما باشد. (۱ نمره)

وضعیت تاهل یک کارمند - تعداد افتاده های درس ریاضی یک کلاس

۲۴- کدام یک از نام های متغیر زیر غیر مجاز است؟ چرا؟ (۱ نمره)

۲code void last name class_code wood-length

سوال کوته پاسخ

۲۵- در یک دستگاه خودپرداز ATM ورودی، خروجی و پردازش را تعیین کنید. (۱ نمره)

۲۶- یک نمونه محیط تولید برنامه متمرکز، نام ببرید (۰,۵)

۲۷- ابزارهای مورد نیاز برای نوشتن و اجرای یک برنامه به زبان سی شارپ را نام ببرید. (۰,۵ نمره)

۲۸- دستوری بنویسید که صفحه را پاک کند (۰,۵ نمره)

۲۹- دو مزیت مهم تایپ برنامه در ویژوال استودیو نسبت به سایر ویرایشگرهای متن، مانند notepad را بنویسید. (۱ نمره)

۳۰- دو مورد از زبانهایی که ویژوال استودیو از آنها پشتیبانی می کند را نام ببرید(۰,۵ نمره)

۳۱- دستوری بنویسید که صدایی با فرکانس ۱۰۰۰ هرتز را به مدت ۳ ثانیه تولید کند. (۰,۵ نمره)

برنامه نویسی

۳۲- برنامه ای بنویسید که نام شما را با رنگ آبی روی پس زمینه سفید، نمایش دهد. (۱ نمره)

۳۳- برنامه ای بنویسید که پیام خوشامدگویی برای شما نمایش دهد. نام کلاس welcome و نام فضای نامی computer باشد. (۱ نمره)

۳۴- برنامه ای بنویسید که نمره پنج درس تخصصی یک دانش آموز را دریافت کرده و معدل آن را نمایش دهد.(۲ نمره)

ارتباط تمارین فصل و اهداف رفتاری

ردیف	اهداف رفتاری فصل چهارم	خودآزمایی	تمرینات برنامه نویسی
۶.	متغیر را تعریف کند و انواع متغیر را در برنامه های خود به کاربندد.	۲و۴و۱۶	
۷.	انواع داده ها را نام ببرد و تفاوت کاربرد هر یک را توضیح دهد.	۵و۶	
۸.	میزان حافظه و محدوده انواع داده ها را بیان کند.	۶ و ۹و۱۱	
۹.	متغیرها را به طور صحیح در برنامه اعلان کند و آن ها را مقداردهی نماید.	۱۰	۱و۲
۱۰.	شکل نمایش نقطه شناور را توضیح دهد و اعداد اعشاری را در این قالب بنویسد.	۱۳ و ۱۴ و۱۵	
۱۱.	ازمتد ReadLine برای دریافت داده های یک برنامه از ورودی استفاده کند.		۱و۲
۱۲.	بر روی رشته دریافتی از ورودی، تغییراتی داده و سپس نمایش دهد.		۱و۲
۱۳.	از متد Parse برای تبدیل یک رشته به یک عدد استفاده کند.		۱

بودجه بندی فصل ها

فصل چهارم
هفته سوم بهمن ماه

موضوع : آشنایی با انواع داده ها و متغیرها

هدف کلی: آشنایی با نحوه معرفی متغیر و انواع داده های عددی، میزان فضا و محدوده آنها

ساعات تئوری: ۳ ساعات عملی: ۵

فعالیتهای آموزشی موازی

-بررسی و حل تمرینهای فصل سوم
-مرور واژگان
- استفاده از MSDN (help) برای نمایش انواع داده ای
-اشکال یابی از برنامه ای با تعاریف غلط متغیرها

فعالیتهای رقابتی و ارزشیابی

- ارزشیابی سه فصل اول
- حل سوالات چندگزینه ای به صورت گروهی رقابتی

آموزش فصل ۴

انواع داده

صفحات ۶۰-۶۴

مدرس:

خانم زهرا باشکوه

شهر کرج

استان البرز

زمستان ۱۳۹۴


Decimal
Long
String
Double
Byte
Float
Int


Long
Int
Byte

فصل ۵

عبارت های محاسباتی

اهداف رفتاری

عملونـد، عملگـر و عبـارت محاسـباتی را تعریـف کنـد و آنهـا را بـه درسـتی بـه کار گیـرد.

عملگرهـای ریاضـی را نـام ببـرد و در عبـارت محاسـباتی از آنهـا اسـتفاده کنـد.

حاصـل عبـارت محاسـباتی را بـا اسـتفاده از جـدول تقـدم عملگرهـا بـه دسـت آورد.

کاربـرد عملگرهـای افزایشـی، کاهشـی و انتسـاب را بیـان نمایـد.

عبارت های محاسباتی

کلید واژه ها: عملگرهای محاسباتی، عملوند،عبارت محاسباتی،اشتراک پذیری عملگرها،تبدیل انواع داده ای

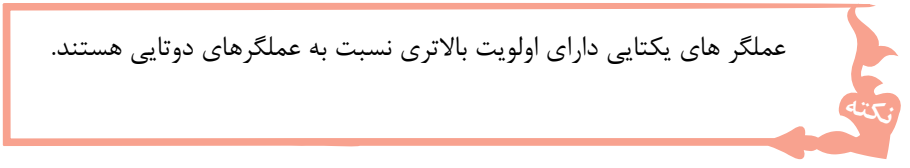
مقدمه

نقش عبارات محاسباتی در هر خط از برنامه به وضوح دیده می شود. هر خط از برنامه به نوعی دارای یک یا چند عبارت خواهد بود؛ بنابراین شناخت این مبحث برای برنامه نویس بسیار ضروری است. عدم آشنایی کافی با موضوعات مطرح شده در این فصل مانند انواع،کاربرد و اولویت عملگرها می تواند باعث خطاهای منطقی یا زمان اجرا و کامپایل برنامه شود.

۵-۱ عملگر

یک عبارت ازتعدادی عملگر و عملوند تشکیل شده است و دارای یک حاصل یا نتیجه می‌باشد. نتیجه یا حاصل یک عبارت ممکن است عددی یا غیرعددی باشد.
در یک عبارت محاسباتی نقش عملگرها به عنوان مجری عملیات، اساسی است. بنابراین نوع، شرکت پذیری و تقدم عملگرها باید به عنوان سه ویژگی اصلی به خوبی مورد بررسی و توجه قرار گیرد.
برخی عملگرها **یکتایی** هستند بدین معنا که روی یک عملوند عمل می کنند.

بر خی نیز میان دو عملوند قرار گرفته و عملیاتی را بین آن دو صورت می دهند. نوع این عملگرها **دوتایی** است. همچنین اگر یک عملگر دارای سه عملوند باشد یک عملگر **سه تایی** خواهد بود. جدول ۱-۵ انواع عملگرها را نشان می‌دهد.



جدول۵-۱ : انواع عملگر

نوع عملگر	نمونه عملگرها
یکتایی	- (قرینه) ،++(افزایشی)، -- (کاهشی)،size of (تعیین اندازه) ، * (آدرس دهی)
دوتایی	+ (جمع)، - (تفریق)، *(ضرب) ، / (تقسیم)، % (باقی مانده تقسیم)،& (and) ،|(or)، ^(or انحصاری (xor))
سه تایی	?: (عملگر شرطی)

نکته

در زبان های شبه C مانند C# عملگر سه تایی به صورت عبارت شرط?:است.

مثال:

```
x>10 ? result=”ok” : result=”not  ok”;
```

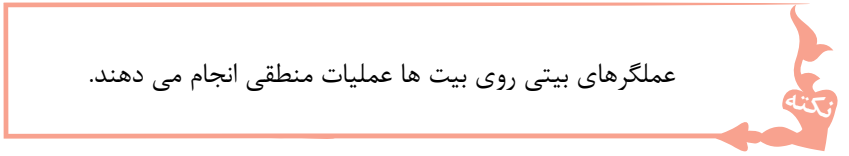
در کد بالا اگر x بزرگتر از ۱۰ باشد به result مقدار ok می دهد و در غیر این صورت به result مقدار not ok را می‌دهد.

۵-۱-۱-ترتیب عملگرها

از دیگـر ویژگـی هـای مهـم عملگرهـا اولویـت آنهـا در عبـارات محاسـباتی اسـت. بدیـن ترتیـب مشـخص مـی نماییـم ترتیـب اجـرای عملیـات چگونـه خواهـد بـود.
در جدول زیر نمونه ای از عملگرها و اولویت آن ها را مرور می کنیم:

جدول ۵-۲ : اولویت عملگرها

عملگر ها	توضیح
() , [] , .	فراخوانی توابع،دستیابی به عضوی از آرایه ها یا کلاس، پرانتز
! ~ - * & sizeof type cast ++x --x	بیشتر عملگرهای یکتایی
* / %	باقی مانده، تقسیم، ضرب
- +	جمع و تفریق
>> <<	شیفت چپ و راست بیتی
^	xor بیتی
|	or بیتی
= += -= *= /= ?: %= &= |= ^=	عملگرهای سه تایی و انتسابی
,	کاما



به عنوان مثال:

نتیجه AND بیتی میان ۲۵ و ۱۰ برابر ۸ می شود	نتیجه OR بیتی میان ۲۵ و ۱۰ برابر ۲۷ می شود	نتیجه XOR بیتی میان ۲۵ و ۱۰ برابر ۱۹ می شود	نتیجه ۲ شیفت به راست عدد ۲۵ برابر ۶ می شود
25: 11001 & 10: 01010 8 : 01000	25: 11001 | 10: 01010 27: 11011	25: 11001 ^ 10: 01010 19 :10011	25: 11001>>2 = 6:00110
مانند جمع و تفریق ریاضی ارقامی که زیر هم قرار می‌گیرند دو به دو عملیات روی آنها انجام می‌شودو فقط اگر هر دو یک باشند حاصل یک است و در بقیه حالات حاصل صفر می‌شود.	مانند جمع و تفریق ریاضی ارقامی که زیر هم قرار می‌گیرند دو به دو عملیات روی آنها انجام می‌شودو فقط اگر هر دو صفر باشند حاصل صفر است و در بقیه حالات حاصل یک می‌شود.	مانند جمع و تفریق ریاضی ارقامی که زیر هم قرار می‌گیرند دو به دو عملیات روی آنها انجام می‌شودو فقط اگر فقط یکی از آنها یک باشند حاصل یک است و در بقیه حالات حاصل صفر می‌شود.	از سمت راست دو رقم را حذف کرده و از سمت چپ دو صفر پشت عدد قرار می‌گیرد.

زبان C# عملگر توان ندارد . میتوان از متدPow کلاس Math که در فضای نام System قرار دارد برای انجام این عمل استفاده نمود. این متد برای محاسبه x^yاستفاده می‌شود.

```
public static double Pow(
        double x,
        double y
)
```

مثال: 100^3 100^3

```
System.Math.Pow(100.00,3.00);
```

۵-۱-۲-عملگر کاما

این عملگر که برای توالی عملیات بکار می رود، در زبان C# به اندازه C گستردگی کاربرد ندارد. در زبانC# عملگر کاما در دستور for را در کتاب درسی دیده اید.

```
for(i=1 , j=2;i<3;i++) …
```

باید توجه داشت که عملگر کاما با کاربرد کاما به عنوان جداکننده در آرگومانهای توابع یا تعریف متغیر ها متفاوت است.

۵-۱-۳-عملگرهای افزایشی و کاهشی

عملگرهای + + و - - عملگرهای افزایشی و کاهشی در زبان C# می باشند. عملگر افزایشی یک واحد به متغیر اضافه می کند و عملگر کاهشی یک واحد از آن کم می نماید.

جدول ۵-۳: عملگرهای افزایشی و کاهشی

عملگر	نشانه	نحوه کاربرد	نوع
افزایشی	++	a++ یا ++a	یکتایی
کاهشی	--	a-- یا --a	یکتایی

بکاربردن این عملگرها قبل یا بعد از متغیر در نحوه اجرا تفاوت ایجاد می کند. اگر عملگر قبل از متغیر بکار رود ابتدا عملیات افزایشی یا کاهشی صورت می گیرد و سپس به متغیر دسترسی پیدا می شود و اگر بعد از آن نوشته شود ابتدا دسترسی به متغیر صورت گرفته و بعد از آن عملیات کاهشی یا افزایشی انجام می شود.

به مثال های زیر توجه کنید:

```
int a=2 , b;
b= ++a * 5 ;  // ++ before  a
```

در کد بالا عملگر ++ قبل از a قرار دارد، بنابراین ابتدا مقدار a برابر ۳ می شود و سپس در ۵ ضرب می شود و مقدار b برابر ۱۵ می شود.

```
int a=2 , b;
b= a++ * 5 ;  // ++ after  a
```

در کد بالا عملگر ++ بعد از a قرار دارد، بنابراین ابتدا مقدار a با مقدار قبلی اش یعنی ۲ در ۵ ضرب می شود و مقدار b برابر ۱۰ می شود و بعد از آن یک واحد به a اضافه می شود و مقدار a برابر ۳ می شود.

به کدهای زیر دقت کنید :

```
int a=2 , b;
b= a * 5 + a++ ;//++ after a :   b->12 (2*5+2)

int a=2 , b;
b= a * 5 + ++a ;// ++ before a :  b->13(2*5+3)
```

۵-۱-۴-شرکت پذیری

شرکت پذیری یک عملگر، ویژگی است که مشخص می کند در شرایط تقدم یکسان، عملیات از چه سمتی گروه بندی و اجرا شود.

بعضی از عملگرها مانند انتساب "**شرکت پذیر از راست**"یا "راست به چپ " هستند.

```
v= 5;

int z = y = x = 9 ;
```

و بعضی عملگرها "**شرکت پذیر از چپ**" یا "چپ به راست " هستند مانند عملگرهای ریاضی . همچنین برخی از عملگرها "**شرکت ناپذیر**" می باشند بدین معنی که رفتار مشخص شده ای در عبارت ندارند که از حوصله این بحث خارج است[۱].

۱. مانند عملگر :- در زبان پرولوگ

۵-۲- تبدیل انواع داده ای

در یک عبارت محاسباتی باید نوع عملوندهای یک عملگر یکسان باشد وگرنه توسط کامپایلر خطای "عدم تطابق داده"[۱] صادر می شود. این نوع خطاها در زبان هایی که نیاز به تعریف متغیر دارد، بیشتر اتفاق می‌افتد. در برخی زبان ها ممکن است عملوندهای ناهمنوع به یکدیگر تبدیل شوند که به آن قالب ریزی نوع داده ای گفته می شود.[۲]این قالب ریزی با تبدیل به دو صورت ضمنی و صریح می تواند انجام شود. تبدیل نوع داده کوچکتر به نوع داده بزرگتر به صورت ضمنی انجام می شود مانند:

```
1) int  a=5; long b=a;
```

متغیر a که از نوع int است به صورت ضمنی به نوع long تبدیل شده است.

```
2) object i=1;
```

در تغییر نوع داده به صورت صریح، برنامه نویس نوع داده ای را که می خواهد داده به آن تبدیل شود، صریحا ذکر می کند.

۱) (نوع داده ای)عبارت;

نتیجه عملیات در این روش، برابر با حاصل عبارت مورد نظر تحت قالب نوع داده ای ذکرشده خواهد بود.

مثال:

```
Console.WriteLine((int)14.6+6);
```

خروجی :

```
20
```

مثال:

```
float number = (float)12.75;
```

عدد 12.75 که از نوع داده double است به صورت صریح نوع داده float تبدیل شده و در متغیر number ذخیره می شود.

۲) عبارت as نوع

عملگر as قابلیت قالب ریزی به انواع مرجع (مانند class) یا nullable[۳] (مانند string) را داراست و در صورت استفاده به جز این دو مورد خطای زمان کامپایل پیش می آید.

---

۱. type mismatch

۲. data type casting

۳. انواعی که قابلیت تهی بودن را دارند

```
object k=”123”;
string str=(k as string) +” is converted to string”;
Console.WriteLine(str);
```

خروجی :

```
123 is converted to string
```

در قطعه کد بالا اگر k=123 مقدار دهی می شد چون 123 یک عدد می باشد و نه یک string بنابراین حاصل k as string برابر با null (تهی) می شد و در نتیجه نمایش به این صورت تغییر می یافت:

is converted to string

همچنین برای تبدیل نوع داده می توان از توابع و متد هایی مانند Parse، ToStringو متدهای کلاس Convert استفاده کرد که عملیات تبدیل را انجام می دهند.
چند نمونه را مرور می کنیم:

متد Parse	
تبدیل رشته متنی ”500“ به int * برای تبدیل به انواع دیگر مانند long،double،bool و... نیز می توان از همین متد استفاده نمود.	System.Console.WriteLine(int. Parse(“500”)*2); خروجی: 1000

متد های کلاس Convert-متد ToChar تبدیل عدد ۶۵ به کاراکتر A *این متد انواع مختلف داده ای را به عنوان ورودی می پذیرد. * برای تبدیل به انواع دیگر می توان از متدهای دیگر کلاس Convert استفاده کرد.	`System.Console.WriteLine(Convert.ToChar(65));` خروجی: A

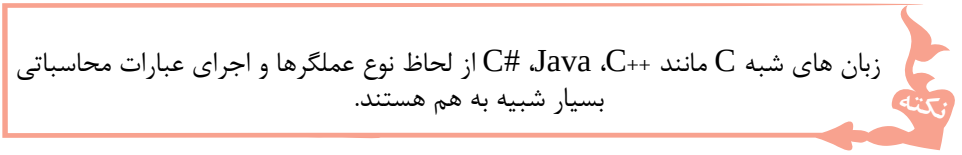
متد TryParse
 	
در این قطعه کد رشته ای توسط ReadLine خوانده می شود. اگر ورودی، قابل تبدیل به int باشد متد TryParse مقدار تبدیل شده را در متغیر i می ریزد و true را به شرط برمی گرداند. بنابراین توان ۲ متغیر i نمایش داده می شود. در غیر این صورت متد TryParse مقدار false را برمی گرداند. سپس پیغام عدد اشتباه، نمایش داده خواهد شد .
 	
* برای انواع دیگر مانند long،double،bool و... نیز می توان از همین متد استفاده نمود.
	`int i;`
	`System.Console.Write("enter a number:");`
	`if(int.TryParse(System.Console.ReadLine(),out i))`
	`System.Console.Write("answer is :{0}",i*i);`
	`else`
	
	`System.Console.Write("wrong input!");`
	
	خروجی: (با ورودی عدد ۴)
	
	enter a number:4
	
	answer is:16
	
	خروجی: (با ورودی غیر عددی)
	
	enter a number:j
	
	wrong number!

قالب ریزی (casting) و تبدیل (conversion) نوع داده ای معمولا در یک معنا به کار برده می شوند. اما در حالت تبدیل ممکن است تغییر نوع صورت نگرفته و فقط یک

نـوع ترجمـه صـورت بپذیـرد. ماننـد متـدی کـه یـک آرایـه را مـی گیـرد و یـک لیسـت را برمـی گردانـد.

۵-۳- مقایسه زبان های مختلف

عملگـر هـا در زبـان هـای مختلـف مـی تواننـد دارای شـکل هـای متفـاوت باشـند و یـا اساسـاً تعریـف نشـده باشـند. در جـدول ۴-۵ مقایسـه ای از سـه زبـان را مشـاهده مـی کنیـد.



زبان های شبه C مانند C++، Java، C# از لحاظ نوع عملگرها و اجرای عبارات محاسباتی بسیار شبیه به هم هستند.

جدول ۵-۴ : مقایسه عملگرها

عملگر	Visual basic	C++	C#
خارج قسمت	\	/ بستگی به عملوندها دارد	/ بستگی به عملوندها دارد
باقی مانده	Mod	%	%
توان	^	ندارد	ندارد
الحاق رشته متنی	& و +	+	+
انتساب باقی مانده	ندارد	%=	%=
انتساب AND بیتی	ندارد	&=	&=
انتساب OR بیتی	ندارد	|=	|=
افزایش	ندارد	++	++

عملگر	Visual basic	C++	C#
کاهش	ندارد	--	--
انتخاب عضو	. !	. ->	.
اندازه متغیر	ندارد	sizeof	sizeof
متمم نسبت به یک	NOT	~	~
المان آرایه	()	[]	[]
AND بیتی	And	&	&
OR بیتی	Or	|	|
XOR بیتی	Xor	^	^

قالب ریزی و تبدیل: در زبانهایی مانند پایتون[1] یا جاوا اسکریپت متغیر ها به صورت خودکار (ضمنی) تعریف می شوند. بنابراین نیازی به عمل قالب ریزی نیست. البته در جایی که عملی برای دو متغیر با دو نوع داده ای متفاوت امکان پذیر نباشد باید عملیات تبدیل صورت پذیرد.

۱. python یک زبان برنامه‌نویسی متن باز،همه منظوره، سطح بالاو شیءگرا است.

واژگان

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	Arithmetical expression	عبارت محاسباتی
۲	Operator	عملگر
۳	Operand	عملوند
٤	Unary operator	عملگر یکتایی
٥	Binary operator	عملگر دوتایی
٦	Ternary operator	عملگر سه تایی
۷	Order of operations	تقدم عملگرها-ترتیب عملگرها
۸	Operator associativity	شرکت پذیری عملگر
۹	Right associative	شرکت پذیر از راست
۱۰	Left associative	شرکت پذیر از چپ
۱۱	Non associative	شرکت ناپذیر
۱۳	Data type casting	قالب ریزی نوع داده ای
۱٤	Data type conversion	تبدیل نوع داده ای

پرسش و پاسخ

۱- برای مقداردهی متغیر در مبنای ۱۶ از روش زیر استفاده می شود:

```
int a = 0xb3  ;
```

آیا می توان متغیر را در مبنای ۸ مقداردهی کرد ؟

خیر - سی شارپ از ثابت عددی مبنای ۸ پشتیبانی نمی‌کند.

۲- آیا انتهای رشته در سی شارپ مثل c با \0 مشخص می شود؟

خیر. انتهای رشته در C# علامت خاصی ندارد.

۳- تفاوت String با S بزرگ و string با s کوچک چیست؟

در سی شارپ تفاوتی بین این دو نیست.
می توان گفت string همان نام System.String است که در C# از آن استفاده می‌شود.

۴- تفاوت عملکرد دو عملگر is و as چیست ؟

عملگر is بررسی می کند که آیا دو متغیر هم نوع هستند یا خیر، در صورتی که هم نوع باشند true بر می‌گرداند.
عملگر as همانند عملیات cast است. اگر امکان تبدیل وجود نداشته باشد به جای اینکه یک استثناء را برگرداند مقدار null را بر می گرداند.

۵- اگر متغیر نوع char در یک عبارت ریاضی استفاده شود، C# با آن چگونه رفتار می کند؟

آن را به عدد تبدیل کرده (تبدیل نوع ضمنی) و در محاسبات شرکت می دهد.
مثال:

```
int m = 65;
char ch = 'A';
m = ch + 1
```

پس از اجرای دستورات مقدار متغیر m عدد ۶۶ خواهد بود.

واحد کار

واحد کار شماره: ۵-۱	فصل ۵ - صفحه ۹۰ تا ۱۰۳	مدت آموزش: ۱۴۰ دقیقه	روش تدریس پیشنهادی: نمایش .پرسش و پاسخ
عنوان: بازی محاسبه عبارات ریاضی (انتخاب مغز ریاضیدان کلاس)		هدف کلی: آشنایی با مفهوم عملگر، عملوند، عملگرهای ریاضی، مفهوم حق تقدم، عبارت محاسباتی، تقدم عملگرها، عملگرهای افزایشی وکاهشی	
اهداف رفتاری: انتظار می رود دانش آموز ۱.مفهوم عملگر، عملوند، عملگرریاضی، حق تقدم، عبارات محاسباتی، عملگرهای افزایشی و کاهشی را توضیح دهد. ۲.عبارت ریاضی ساده و پیچیده را محاسبه و نتیجه را نمایش دهد. ۳.کاربرد این عبارات را در برنامه نویسی شرح دهد. ۴.نوع نتیجه عبارت را تعیین نماید.		اهداف پنهان درس: پی بردن به قدرت وسرعت بالای محاسبه درکامپیوتر، تقویت قوه درک و فهم عبارات ریاضی و عملگرها هدف ایده آل: آموزش اولویت بندی کارها و زمان بندی و برنامه ریزی در زندگی برای رسیدن به هدف و موغقیت.	

فرایند تدریس:

پاکت هایی تهیه می کنیم و انواع داده ای عددی را روی هر پاکت درج می کنیم. برای انواع داده ای بزرگتر، پاکت های بزرگتری انتخاب می کنیم.۶ عملگر اصلی موجود در جدول صفحه ۹۲ را روی برگه هایی چاپ نموده و بر روی سینه ۶ دانش آموز میچسبانیم.این دانش آموزان می توانند از بین افراد کوتاه قد تر انتخاب شوند. با توضیح نوع عملگر(یکتایی و دوتایی) به طرح عبارات محاسباتی می پردازیم. ابتدا از عباراتی با دو عملگر شروع می کنیم. عملوندهای موجود در هر عبارت را روی برگه هایی نوشته یا چاپ می کنیم و به هنرجویانی که می خواهند نقش عملوند داشته باشند می دهیم. از دانش آموزان نقش عملگر و عملوند، می خواهیم به شکلی بایستند که در عبارت قرار دارد. اکنون شروع انجام عملیات را در طول زمان مشخص، اعلام می کنیم. بعد از سپری شدن فرصت هنرجویان باید نتیجه را به همراه نام خود و شماره سوال (عبارت)، در کاغذ کوچکی بنویسند و با تشخیص نوع، در پاکتهای مربوط قرار دهند. سپس از دانش آموزان نقش عملگر می خواهیم به ترتیب اولویت با آمدن روی سکو، عملوندهای خود را صدا بزنند و نتیجه عمل را به عملگر بعدی تحویل دهند. با محاسبه آخرین عملگر، نتیجه به دست خواهد آمد. این بازی را برای عبارات محاسباتی با تعداد بالاتر عملگر هم انجام می دهیم در پایان با محاسبه ۱۰ عبارت، چند دانش آموز داوطلب، جواب ها را از پاکت ها بیرون می آورند و به ترتیب سوال بررسی می کنند. به ازای هر جواب صحیح که داده اند نشان++ (عملگر افزایشی) می دهیم. همچنین از همه می خواهیم زمانیکه جواب هر سوال مربوط به خودشان بررسی شد یک علامت – (عملگر کاهشی) بگذارند تا با کاستن از تعداد سوالات، مشخص شود چه تعداد جواب بررسی نشده دارند. در انتها با شمارش نشان ها، مغز ریاضیدان را پیدا می کنیم.

واحد کار شماره : ۵-۱	فصل ۵ - صفحه ۹۰ تا ۱۰۳

توصیه های اجرایی:

- جدول نوع عملگر، نشانه، نام عملگر، اولویت، نوع عبارت و مثال را روی مقوای بزرگ ترسیم کنید و روی دیوار نصب کنید.
- به همه دانش آموزان فرصت برابر شرکت در مسابقه و کسب امتیاز بدهید. برای اینکار از دانش آموزان نقش عملگر و عملوند هم بخواهید نتیجه را محاسبه کنند.
- در مواردی با راهنمایی غلط ، فرصت تصحیح خطا را برای دانش آموزان فراهم کنید.
- ملاک طراحی عبارت ها باید کار با عملگرها باشد و از دادن عملوندهایی که کار محاسبه را مشکل می کنند پرهیز شود.

ارزشیابی از انتظارات:

۱. -4+2/8 ؟
۲. 5*(17+ 4- % 7) ؟
۳. 5*5%5*122 ؟
۴. 17+2*5-4/10 ؟
۵. –(4- *10-+5/2*-20) ؟

تکالیف و پروژه های پیشنهادی:

آیا می خواهید بدانید که از لحظه تولد تا الان چند ساعت، چند دقیقه و یا چند ثانیه زندگی کرده اید؟
برنامه را به این شکل تحلیل، طراحی و کد نویسی نمایید که با توجه به نوع متغیر و مقدار داده ها عدد سن شما را به سال، ماه و روز دریافت نموده و سن شما را به ساعت، دقیقه و ثانیه نمایش دهد؟
راهنمایی : هر سال برابر است با 365*24*60*60 ثانیه

نمونه سوال از فصل ۵

درستی یا نادرستی هر عبارت را تعیین کنید. **د** برای درست و **ن** برای نادرست.

۱- با پرانتز می‌توان اولویت عملگر ها را تغییر داد.

۲- حاصل یک عبارت از نوع اعشاری را می‌توان به یک متغیر نوع صحیح انتساب داد.

۳- عملگرهای دوتایی شرکت پذیر راست هستند.

۴- عملگر ضرب و تقسیم دارای اولویت یکسان و پایین تر از عملگر جمع و تفریق هستند.

۵- در پرانتزهای تودرتو ابتدا داخلی ترین پرانتز اجرا می‌شود.

۶- مقدار متغیری از نوع float را می توان در متغیری از نوع double ذخیره کرد.

۷- قسمت اعشار مقدار ذخیره شده در متغیر double باید ۱۵ رقم باشد.

در سوالات زیر گزینه صحیح را انتخاب کنید.

۸-در عبارت روبرو کدام عملگر اول اجرا می شود؟ 23- 10*4/5%3

الف) ٪ ب) / ج) - د)*

۹-کدام گزینه صحیح است؟

الف) اولویت عملگر * از عملگر قرینه بیشتر است.

ب) اولویت عملگر / از عملگر ٪ بیشتر است.

ج) اولویت عملگر ٪ از عملگر + بیشتر است.

د) اولویت عملگر تفریق از عملگر / بیشتر است.

۱۰- اگر number=11 نتیجه اجرای دستور زیر چیست؟

```
Console.WriteLine(++number * 3-2);
```

الف) 33 ب) 11 ج) 12 د)34

۱۱- اگر در برنامه یا عبارتی بخواهیم عدد اعشاری با دقت حداکثر ۷ رقم ذخیره کنیم پس از هر عدد اعشاری چه حرفی باید قرار دهیم؟

الف) f یا F　　ب) h یا H　　ج) u یا U　　د) d یا D

در جملات زیر، جاهای خالی را با عبارات مناسب پرکنید.

۱۲-عملگر انتساب شرکت پذیر ..... است. (چپ – راست)

۱۳-عملگرهای ++ و – از عملگرهای نوع ..... هستند. (یکتایی – دوتایی)

۱۴-حاصل تقسیم یک عدد اعشاری بر یک عدد صحیح، یک عدد از نوع ..... است.

با استفاده از عملگرهای انتساب به سوالات زیر پاسخ دهید. چه عملگری مورد نیاز است.

۱۵- نصف کردن مقدار یک متغیر

۱۶- افزایش ۸ واحد به یک متغیر

۱۷- پنج برابر کردن مقدار یک متغیر

۱۸- کاهش ۴ واحد از مقدار یک متغیر

۱۹- با توجه به تعریف متغیرهای a و b خطای هریک از دستورات زیر را مشخص کنید.

```
byte b = 50;
int a=10;
```

الف) b = b + a;
ب) a /= 1.0;
ج) b = -7;

۲۰- خروجی قطعه کد زیر را به دست آورید ؟

```
int x, y;
x = 5;
y = x++;
y = ++x;
x = y--;
Console.WriteLine(“x:={0} y:={1}”,x,y);
```

۲۱- پیام خطای هر قطعه کد را با نوشتن شماره پیام، جلوی قطعه کد مشخص کنید. یک پیام خطا اضافه است.

1	Literal of type double cannot be implicitly converted to type 'float'; use an 'F' suffix to create a literal of this type
2	Constant value '-45897' cannot be converted to a 'ushort'
3	Constant value '1' cannot be converted to a 'bool'
4	Cannot implicitly convert type 'ushort' to 'bool'

الف) `bool c = 1;` (....)

ب) `float f = 45.87;` (....)

ج) `ushort b = -45897;` (....)

ارتباط تمارین فصل و اهداف رفتاری

ردیف	اهداف رفتاری فصل پنجم	خودآزمایی	تمرینات برنامه نویسی
۱.	عملوند، عملگر و عبارت محاسباتی را تعریف کند و آنها را به درستی به کارگیرد.	۱ و ۲	۱تا ۱۱
۲.	عملگرهای ریاضی را نام ببرد و در عبارت محاسباتی از آنها استفاده کند.	۴و ۶ و۷	
۳.	حاصل عبارت محاسباتی را با استفاده از جدول تقدم عملگرها به دست آورد.	۲و۵	
۴.	کاربرد عملگرهای افزایشی، کاهشی و انتساب را بیان نماید.	۶ و ۱۱ و ۱۲	

فصل چهارم
هفته اول اسفند ماه

موضوع : عبارت های محاسباتی

هدف کلی: معرفی و شناخت عملگر، عملوند و عبارت محاسباتی

ساعات تئوری: ۵ **ساعات عملی:** ۳

فعالیتهای آموزشی موازی

-بررسی و حل تمرینهای فصل چهارم
-مرور واژگان
-طراحی پوستر عملگرها و اولویت های آن به وسیله ی هنرجویان

فعالیتهای رقابتی و ارزشیابی

- آزمون ۳ فصل اول
-مسابقه دونفری از عملگرها مطابق پوستر نصب شده

بودجه بندی فصل ها

فصل پنجم
هفته دوم اسفند ماه

موضوع : عبارت های محاسباتی

هدف کلی: بررسی وتقدم عملگرها -کاربرد عملگرهای افزایشی و کاهشی

ساعات تئوری: ۲ ساعات عملی: ۶

فعالیتهای آموزشی موازی
-بررسی و حل تمرینهای فصل پنجم
- نمایش خطاهای منطقی یک برنامه دراثر تعریف اشتباه عبارت

فعالیتهای رقابتی و ارزشیابی
- ارزشیابی فصل چهارم
- طرح پرسش از گروه رقیب(عبارت های محاسباتی با اعداد تک رقمی توسط گروهی از دانش آموزان برای گروه دیگر)

فصل ۶

دستورهای شرطی

اهداف رفتاری

انواع عملگرهای مقایسه ای را توضیح دهد و آنها را در برنامه استفاده کند.

ساختار دستور if و if_else را در برنامه های شرطی خود استفاده کند.

مفهوم عملگرهای منطقی را شرح دهد و انواع آنها را نام ببرد.

از عملگرهای منطقی به طور صحیح در برنامه ها استفاده نماید.

با استفاده از جدول عملگرهای منطقی، نتیجه عبارتهای ترکیبی منطقی و مقایسه ای را به دست آورد.

اولویت عملگرها را با استفاده از جدول تقدم عملگرها در حل عبارات ترکیبی به کار بندد.

دستور switch را در برنامه های خود به کار گیرد.

با ترکیب دستورهای شرطی، برنامه هایی با ساختار پیچیده شرطی بنویسد.

دستورهای شرطی

کلیـد واژه هـا : عبـارت منطقـی، سـاختار شـرطی، دسـتور if، دسـتور if-else، دسـتور if-else پیچیـده، دسـتورswitch

مقدمه

یکی از راه کارهای حل مسایل در برنامه نویسی استفاده از ساختارهای تصمیم گیری است تا بر اساس بررسی یک شرط، دستورات مورد نیاز برای رسیدن به حل مساله اجرا شود.

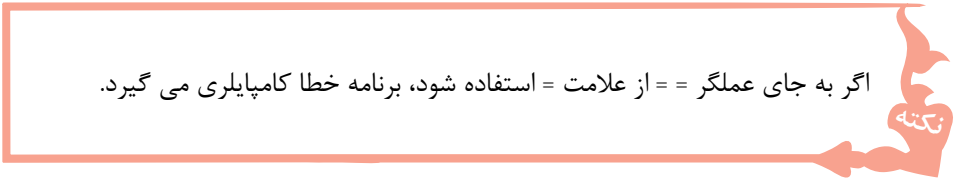
در این فصل با دو ساختار شرطی if و switch آشنا می شویم.

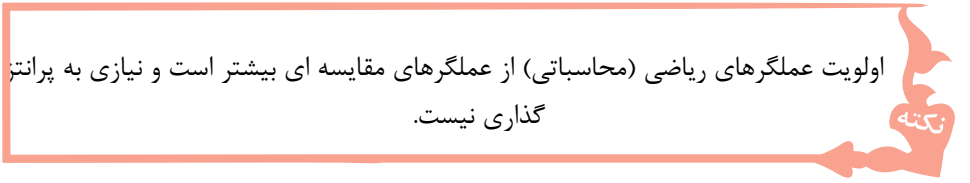
۶-۱- عبارت منطقی

در دسـتورات شـرطی، ابتـدا یـک عبـارت منطقـی بررسـی مـی شـود و بـر اسـاس حاصـل آن، دسـتورات مـورد نیـاز نوشـته مـی شـود. در عبـارت منطقـی معمـولا از عملگرهـای مقایسـه ای اسـتفاده مـی شـود کـه حاصـل آن true (درسـت) یـاfalse (نادرسـت) اسـت و بـر اسـاس آن تصمیـم گیـری انجـام مـی شـود. در زبـان C# اگـر حاصـل ایـن عبـارت، true یـا false نباشـد در زمـان کامپایـل خطـا ایجـاد مـی شـود امـا در زبـان هایـی ماننـد C اگـر یـک عبـارت دیگـر جایگزیـن شـود خطایـی حـادث نمـی شـود چـرا کـه هـر عبـارت بـا نتیجـه یـا مقدارغیـر صفـر بـه عنـوان true و عبـارت بـا نتیجـه یـا مقـدار صفـر بـه عنـوان false تلقـی خواهـد شـد.

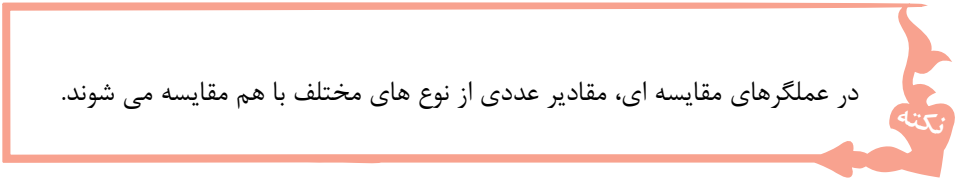
بـا توجـه بـه همیـن موضـوع، در زبـان C# اگـر مـا بـه جـای عملگـر = = بـرای بررسـی تسـاوی دو عبـارت، از عملگـر = کـه عملگـر انتسـاب اسـت اسـتفاده کنیـم در زمـان کامپایـل خطـا داده مـی شـود. در حالـی کـه همیـن موضـوع در زبـان C و در زمـان کامپایـل خطـا ایجـاد نمـی کنـد امـا باعـث خطـای منطقـی برنامـه مـی شـود.

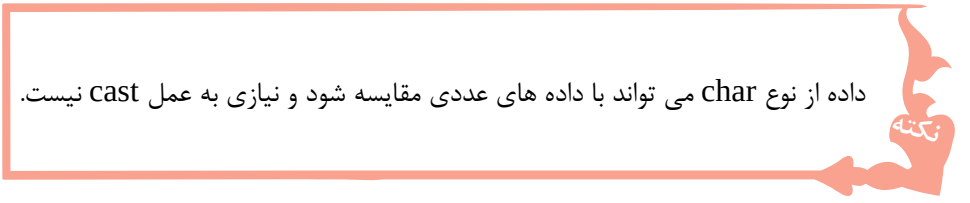
یکـی دیگـر از مـواردی کـه در عبـارات منطقـی بایـد ماننـد عبـارات محاسـباتی بـه آن توجـه شـود، نـوع داده ای عملونـدهـای شـرکت کننـده در عبـارت منطقـی اسـت! اگـر عملونـدهـای داده ای ناهمگونـی در عبـارت منطقـی وجـود داشـته باشـند و تبدیـل ضمنـی بیـن ایـن عملونـدهـا صـورت نگیـرد بایـد تبدیـل صریـح صـورت گیـرد .هنـگام کاربـرد عملگرهـای مقایسـه ای بـه نـکات زیـر توجـه کنیـد.





مثال: در عبارت ۴ = > ۲ + m ، ابتدا مقدار m با عدد ۲ جمع می شود و سپس عمل مقایسه انجام می شود.





(در صورتی که در عبارات محاسباتی و انتساب باید حتما از عمل cast استفاده شود.)

با توجه به نکات بالا به جدول۶-۱ دقت کنید.

جدول۶-۱ : حالات مختلف استفاده از عملگر = =

مقادیر اولیه	شرط	خطا	نتیجه
int m = 66; float n = 66f;	m = = n	ندارد	true
int m = 65; char ch = ‘A’;	m = = ch	ندارد	true
int m = 66; char ch = ‘A’;	m = = ch + 1	ندارد	true
char ch = ‘A’;	ch = = 65	ندارد	true
float n = 65f; char ch = ‘A’;	n = = ch	ندارد	true
int m = 66; string str = “string”;	m = =str	دارد	

در مثال آخر چون انواع داده ای string و int قابلیت تبدیل ضمنی به هم را دارا نمی باشند، خطا ایجاد می شود. اما در مورد دوم به دلیل آنکه نوع char و int به صورت ضمنی تبدیل به یکدیگر می شوند، بدون داشتن هیچ خطایی نتیجه true برای عبارت حاصل می شود.

۶-۲-ساختارهای شرطی

۶-۲-۱-ساختار if و if-else

رایج ترین دستور شرطی که در زبان های مختلف کاربرد دارد دستور if می باشد. این دستور را می توان به شکل ساده if و یا if – else و یا به شکل if های تودرتو استفاده نمود. در جدول زیر شکل های کلی استفاده از این دستور در زبان C# را مشاهده می کنید.

جدول۶-۲ : شکل های کلی دستور if

if(عبارت منطقی)
{ دستورات }
else if(عبارت منطقی)
{ دستورات }
else
{ دستورات }
	if(عبارت منطقی)
	{ دستورات }
	else
	{ دستورات }
		if(عبارت منطقی)
		{دستورات}

نکته

بعد از علامت پرانتز نباید کاراکتر ; (semicolon) گذاشت. در این حالت برنامه خطای کامپایلری ندارد و اجرا می شود ولی معادل دستور زیر می شود:

if(عبارت منطقی) ; معادل if (عبارت منطقی)
;
دستور ;

یـن بـدان معناسـت کـه اگـر شـرط برقـرار باشـد، هیـچ دسـتوری اجـرا نمـی شـود چـون قبـل از ; دسـتوری نیسـت. و دسـتور if تمـام شـده اسـت و سـپس خـط بعـدی یعنـی «دسـتور» انجـام مـی شـود. پـس حتـی اگـر شـرط برقـرار نباشـد « دسـتور» انجـام مـی شـود و ایـن منطـق برنامـه را تغییـر مـی دهـد و خطـای منطقـی رخ مـی دهـد.

مثال:

دستوری بنویسید که اگر مقدار نمره بزرگتر مساوی ۱۲ است، پیام قبولی را چاپ کند.

```
Mark = 8;

if( Mark >= 12)

    Console.WriteLine(“passed”);
```

در کد بالا چون نمره کمتر از ۱۲ است، دستور if انجام نمی شود و مقداری چاپ نمی شود و منطق برنامه نیز درست است.

ولی اگر داشته باشیم :

```
Mark = 8;

if( Mark >= 12) ;

    Console.WriteLine(“passed”);
```

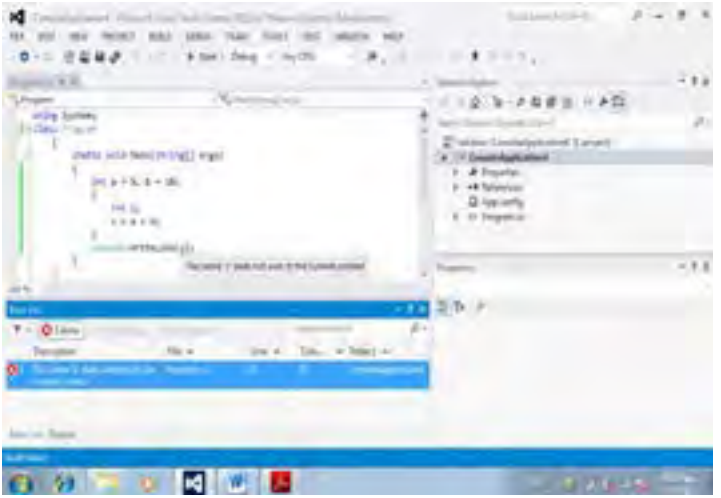
با گذاشتن ; بعد از پرانتز if، دستورif پایان یافته است و دستور چاپ ارتباطی با دستور if ندارد و در هر حال اجرا می شود.

استفاده از بلاک دستورات (قرار دادن دستورات میان آکولادهای باز و بسته {}) در صورتی لازم است که چندین دستور را بخواهیم برای if یا else تعیین نماییم اما اگر بخواهیم تنها یک دستور را برای این قسمت ها مشخص کنیم نیازی به آن نیست.

در واقع بلاک دستورات، یک بخش منطقی مجزا را در میان دستورات فراهم می کند. بنابراین باید دقت شود که متغیرهای تعریف شده در محدوده بلاک، بیرون از آن قابل دستیابی نیستند.

مثال :

به برنامه و پیغام خطای شکل ۶-۱ دقت کنید:



شکل ۶-۱

یعنی عبارت S در این قسمت شناخته شده نیست. چون متغیر S در داخل بلاک تعریف شده و بعد از تمام شدن بلاک، متغیر S هم از بین می رود. برای رفع این خطا به شکل زیر برنامه را تصحیح می کنیم:

```
using System;
class Program
  {
    static void Main(string[] args)
    {
      int a = 5, b = 10;
      int s;
      {
        s = a + b;
      }
       Console.WriteLine(s);
    }
  }
```

زمانی که بخواهیم دو یا چند عبارت منطقی را با هم ترکیب کنیم از عملگرهای منطقی استفاده می کنیم. انواع عملگرهای منطقی در جدول ۶-۳ آمده است.

جدول ۶-۳: انواع عملگرهای منطقی

نام عملگر	نشانه
نقیض	!
و	&&
یا	||
یا انحصاری	^

نکته

یا انحصاری (^) اگر روی عملوندهای Boolean عمل کند حاصل منطقی (true یا false) خواهد داشت در غیر اینصورت اگر روی اعداد صحیح عمل نماید یک مقدار صحیح را نتیجه می دهد و یک عملگر منطقی محسوب نمی شود.

مثال : می خواهیم بررسی کنیم مقدار متغیر x بین ۱۰ و ۲۰ هست یا خیر؟ درریاضی این عبارت به شکل ۲۰ > x > ۱۰ نمایش داده می شود ولی در اینجا داریم :

x> 10 && x < 20

نکته

در پردازش عبارات، اولویت عملگر ها به ترتیب عبارت است از:
عملگر ریاضی – عملگر مقایسه ای – عملگر منطقی. در نتیجه در استفاده از عملگر منطقی نیازی به گذاشتن پرانتز نیست.

نکته

در عملگر && اگر ارزش عبارت سمت چپ false باشد دیگر ارزش عبارت سمت راست بررسی نمی شود.

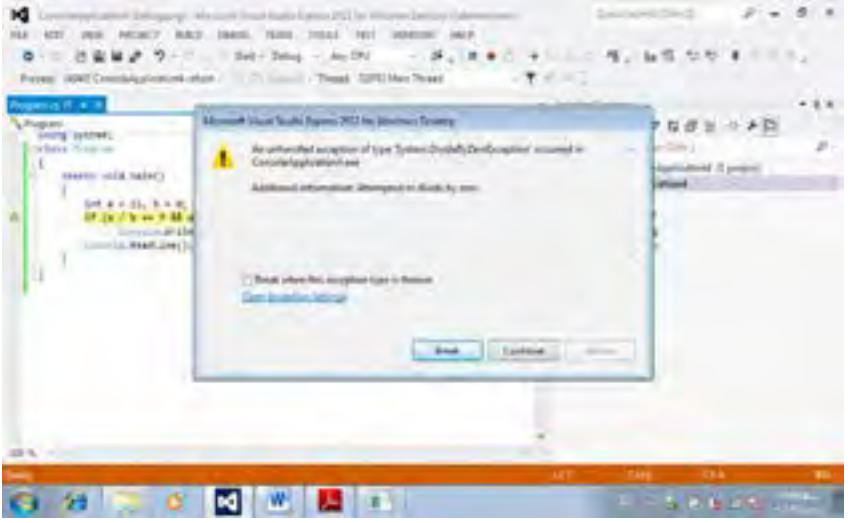
نکته

در عملگر || اگر ارزش عبارت سمت چپ true باشد دیگر ارزش عبارت سمت راست بررسی نمی شود.

مثـــال: در برنامــه زیــر در اجــرای عبــارت منطقــی ســمت چــپ عملگــر &&، خطــای زمــان اجــرای «تقســیم بــر صفــر» روی مــی دهــد.

```
using System;
class Program
{
   static void Main()
   {
      int a = 15, b = 0;
      if (a / b == 9 && a < 10)
          Console.WriteLine(“hello”);
      Console.ReadLine();
   }
```

پیغام خطا در زیر نمایش داده شده است:



شکل۶-۲: خطای تقسیم بر صفر

حــال اگــر برنامــه را بــه شــکل زیــر تغییــر دهیــم. یعنــی جــای دو عبــارت منطقــی را عــوض کنیــم، بــه شــکل زیــر:

```
if (a < 10 && a / b == 9)
```

ابتــدا عبــارت ســمت چــپ بررســی مــی شــودکه ارزش آن false اســت و چــون در عملگــر && وقتــی یکــی از عملوندهــا false باشــد، حاصــل آن false مــی شــود، در نتیجــه عملونــد دوم

اجـرا نمـی شـود. پـس در اینجـا دیگـر خطـا روی نمـی دهـد، چـون اصـلا اجـرا نمـی شـود .

۶-۲-۲- ساختار if-else پیچیده

ایـن سـاختار زمانـی بـکار مـی رود کـه مـی خواهیـم چندیـن شـرط مرتبـط بـه هـم را پشـت سـرهم بررسـی کنیـم.

اگر عبارت منطقی ۱ درست نباشد، عبارت منطقی ۲ بررسی می شود.

```
if (عبارت منطقی۱)
{
   دستورات شماره ۱ ;
}
else if (عبارت منطقی۲)
{
    دستورات شماره ۲ ;
}
else if (عبارت منطقی۳)
{
    دستورات شماره ۳ ;
}
else
{
   دستورات شماره ۴ ;
{
```

شکل ۶-۳: ساختار if-else پیچیده

آیـا مـی تـوان بجـای if-else از چندیـن دسـتور if اسـتفاده نمـود؟ چـرا ؟چـه خطایـی ممکـن اسـت رخ دهـد ؟

نتایج دو برنامه زیر را با هم مقایسه کنید:

مـی خواهیـم قطعـه برنامـه ای بنویسـیم کـه بـر اسـاس مقـدار قیمـت کالا، درصـد سـود متناسـب بـا آن در نظـر گرفتـه شـود و قیمـت آن محاسـبه شـود. مثـلا بـرای اجنـاس زیـر ۱۰۰ هـزار تومـان، ۲۰ درصـد و بیـن ۱۰۰۰۰۰ تـا ۲۰۰۰۰۰، ۵۱ درصـد سـود در نظـر گرفتـه شـود.

اگـر قیمـت کالا ۹۰۰۰۰ تومـان باشـد و برنامـه را بـه دو روش الـف و ب بنویسـیم، نتایـج متفـاوت هستند .

الف)

```
float price;

Console.Write("Enter price : ");
price = float.Parse(Console.ReadLine());

 if (price <= 100000)
      price = price + price * 0.2f;
 else if (price > 100000 && price <= 130000)
      price = price + price * 1.5f;

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

شکل ۶-۴ : خروجی قطعه برنامه الف

ب)

```
float price;


Console.Write(“Enter price : “);
price = float.Parse(Console.ReadLine());

if (price <= 100000)
   price = price + price * 0.2f;
if (price > 100000 && price <= 130000)
   price = price + price * 0.15f;

Console.WriteLine( “price = “+price);
```

شکل ۶-۵ : خروجی قطعه برنامه ب

نتایج این دو قطعه برنامه، متفاوت هستند. این به دلیل خطای منطقی برنامه روش ب است .

if اول روی مقدار متغیر price تغییرات ایجاد می کند و این باعث بروز خطای منطقی در if دوم می شود.

در این برنامه حتما باید از else if استفاده نمود.

۶-۲-۳- ساختار switch

این دستور، ساختاری فراهم می کند که بر اساس مقادیری که یک عبارت می تواند داشته باشد دستورات خاصی اجرا شوند.

```
switch (عبارت)
{
   case ۱ مقدار:
      ;دستور۱
      break;
   case : ۲مقدار
      ;دستور۲
      break;
   .
   .
   .
   default:
      ;دستورهای دیگر
      break;
}
```

ساختار کلی دستور switch

شکل ۶-۶ : ساختار کلی دستور **switch**

هر کدام از case ها برای حالت یا مقدار خاص عبارت، دستوراتی را برای اجرا معین می کند. با استفاده از کلمه کلیدی default می توان دستوراتی را برای حالت پیش فرض مشخص کرد تا در هنگامی که مقدار عبارت با هیچ کدام از مقادیر معین شده توسط case ها، برابر نبود، آن دستورات اجرا گردند.

توجه: مقدار درج شده توسط case باید **ثابت** باشد.بنابراین از متغیرها نمی توان استفاده نمود و یا مانند زبان VB (در دستور select case)نمی توان از محدوده مقادیر استفاده نمود.

همچنین در دستورات مربوط به هر حالت باید از یک دستور پرش استفاده شود که به صورت رایج از break استفاده می شود. این دستور اجرا را به بیرون از بلاک switch منتقل می کند. با استفاده از دستور goto نیز می توانیم اجرا را به خارج از switch و یا به دستورات یک case منتقل کنیم. دستور goto باعث انتقال اجرا به یک برچسب می شود.

مثال: برنامه ای می خواهیم که با توجه به ایستگاه های متروی شهر تهران(خط

یـک)، ایسـتگاه هـای بعـدی تـا ایسـتگاه دکتـر شـریعتی را بـا توجـه بـه ایسـتگاه فعلـی نمایـش دهـد. فـرض مـی کنیـم ایسـتگاه ابتدایـی ایسـتگاه مصلـی باشـد.

شکل ۶-۷

```
using System;
class Program
{
   static void Main()
   {

   input: Console.Write(“enter current station:”);
      switch (Console.ReadLine())
      {
         case “mosalla”:
            Console.WriteLine(“shahid hemmat”);
            goto case “shahid hemmat”;
         case “shahid hemmat”:
            Console.WriteLine(“shahid haghani”);
            goto case “shahid haghani”;
         case “shahid haghani”:
            Console.WriteLine(“mirdamad”);
            goto case “mirdamad”;
```

```
        case “mirdamad”:
            Console.WriteLine(“doctor shariati”);
            break;
        default:
            Console.WriteLine(“enter true station”);
            goto input;
    }
    Console.ReadKey();
  }
}
```

شکل ۶-۸

یادآوری می شود که در زبان C++، اگر دستور پرش در دستورات یک case نوشته نشود خطا ایجاد نمی شود و دستورات case بعدی اجرا می گردد.

سوال

چگونه می توان عملگر || را با استفاده از دستور switch شبیه سازی کرد؟

نکته

زبان C# اگر درون قسمت دستورات یک case هیچ دستوری نوشته نشود دستورات case بعدی اجرا می شود.

مثال: در این قطعه برنامه به ازای مقادیر ۱ یا ۲ یا ۳ عبارت low و به ازای مقادیر ۴ یا ۵ یا ۶ عبارت middle و به ازای مقادیر ۷ یا ۸ یا ۹ عبارت high نمایش داده می شود.

```
int a;
a = int.Parse(Console.ReadLine());
switch (a)
{
    case 1:
    case 2:
    case 3: Console.WriteLine(“low”); break;
```

```
        case 4:
        case 5:
        case 6: Console.WriteLine("middle"); break;
        case 7:
        case 8:
        case 9: Console.WriteLine("high"); break;
      }
```

با استفاده از دستور شرطی if می توان ورودی های برنامه را کنترل کرد. به مثال زیر توجه کنید:

مثال: برنامه ای بنویسید که سال تولد کاربر و سال جاری را از کاربر دریافت کرده (به صورت ۴ رقمی) و سن کاربر را محاسبه کند.

```
using System;
class Program
{
  static void Main()
  {
    int BirthYear, Year , Age;
    Console.Write("Enter your BirthYear (yyyy) : ");
    BirthYear = Int32.Parse(Console.ReadLine());
    Console.Write("\n Enter Year  (yyyy) : ");
    Year = Int32.Parse(Console.ReadLine());
    if (BirthYear < 1000 || BirthYear > 10000 || Year < 1000 || Year > 10000)
      Console.WriteLine("\n \n !!!   You entered Invalid data ");
    else
    {
      Age = Year - BirthYear;
      Console.WriteLine("\n Your age -> " + Age);
    }
    Console.ReadKey();
    }
  }
```

۶-۳-مقایسه در زبان های دیگر

دستور if با مشابهت زیادی در زبانهای مختلف لحاظ شده است:

Visual Basic

```
If  شرط Then
  …
Else
  …
End If
```

matlab

```
if شرط
…
else
…
end
```

شکل۶-۹ : ساختار if درزبان Visual Basicو Matlab

دستور switch در برخی زبانها با ساختار شبیه switch و با همین نام کلیدی به کار برده می شود اما در زبانهایی مانند Visual Basic تفاوت های نحوی و ساختاری بیشتری دارد همچنین در برخی زبان ها مانند python دستوری مشابه طراحی نشده است که می توان آن را با تکنیک های دیگر از جمله if – else if جایگزین کرد.

Visual Basic

```
Select Case عبارت
case حالت۱
 دستورات
case حالت۲
 دستورات
.
.
Case Else
 دستورات
End Select
```

matlab

```
switch عبارت
case حالت۱ :
 دستورات
case حالت۲ :
 دستورات
.
.
Otherwise:
 دستورات
end
```

شکل ۶-۱۰ : ساختار switch درزبان Visual Basicو Matlab

واژگان

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	**Boolean expresion**	**عبارت منطقی**
۲	**Conditional command**	**دستور شرطی**
۳	**Compile error**	خطای کامپایل
۵	**Deciding**	**تصمیم گیری**

پرسش و پاسخ

۱- اولویت عملگرهای ریاضی و مقایسه ای نسبت به یکدیگر چگونه است؟

اولویت عملگرهای ریاضی (محاسباتی) از عملگرهای مقایسه ای بیشتر است و نیازی به پرانتزگذاری نیست.

مثال: در عبارت m + 2 < = 4، ابتدا مقدار m با عدد 2 جمع میشود و سپس عمل مقایسه انجام می شود.

۲- آیا یک داده از نوع char می تواند با داده های عددی مقایسه شود؟

یک داده از نوع داده char می تواند با داده های عددی مقایسه شود و نیازی به عمل cast نیست. (در صورتیکه در عبارات محاسباتی و انتساب باید حتما از عمل cast استفاده شود.)

بنابراین در مقایسه زیر خطایی نخواهیم داشت و نتیجه درست خواهد بود:

```
char a = 'a';

int b = 97;

if (a==b)

 {

   ....

 }
```

۳-اگر متغیر نوع char در یک عبارت ریاضی استفاده شود، C# با آن چگونه رفتار می-کند؟

آن را به عدد تبدیل کرده (تبدیل نوع ضمنی) و در محاسبات شرکت می دهد.

مثال:

```
int m = 65;

char  ch = 'A';

m = ch + 1
```

پس از اجرای دستورات مقدار متغیر m عدد ۶۶ خواهد بود.

۴-عملکرد دستورات زیر چیست؟

if (عبارت منطقی) ;

دستور ;

دستور دارای خطای نحوی نیست ولی خطای منطقی دارد. قرار دادن نقطه ویرگول یعنی پایان یافتن یک دستور، بنابراین اگر شرط برقرار باشد، هیچ دستوری اجرا نمی شود.

این کار نیز باعث بروز خطا نمی گردد.

۵- در مواردی که عبارت منطقی در ساختار شرط، ترکیبی باشد اولویت عملگرها به چه صورت است؟

اولویت عملگر ها به ترتیب عبارت است از : عملگر ریاضی- عملگر مقایسه ای- عملگر منطقی.
در نتیجه در استفاده از عملگر منطقی نیازی به گذاشتن پرانتز نیست.

۶-تفاوت متد Read با Readline چیست؟

متد Read برای دریافت یک کاراکتر از ورودی به کار می رود (در صورتی که چند کاراکتر وارد شود فقط اولین کاراکتر را به عنوان ورودی در نظر می گیرد)

ReadLine برای دریافت یک رشته از ورودی به کار می رود و رشته ای که تا قبل از فشار دکمه Enter وارد شده است را برمی گرداند.

۷- در قطعه کد زیر، اگر i از نوع char تعریف شود خطا می گیرد.

چگونه می توان این خطا را برطرف کرد؟

```
int i = Console.Read();

    Console.WriteLine(i);
```

برای رفع خطا می توان خروجی Read را با Convert.ToChar به char تبدیل کرد.

```
int i = Convert.ToChar (Console.Read());

    Console.WriteLine(i);
```

۸-حوزه یک متغیر در C# چگونه است؟

متغیری که داخل یک بلاک تعریف می‌شود، فقط در داخل آن بلاک و زیربلاکهای آن قابل استفاده است. بنابراین داخل بلاک یا زیر بلاکهای آن نمی توان متغیری

هم نام آن تعریف کرد.

۹-در چه صورت می توان دو متغیر هم نام در یک برنامه داشت؟

زمانی می توان متغیر هم نام در بلاکهای مختلف بکار برد که به شکل دو بلاک مستقل از یکدیگر باشند.

و یا یکی با حروف کوچک و دیگری با حروف بزرگ باشد.

مثال اعلان دو متغیر هم نام با رعایت کردن قوانین محدوده اعلان:

```
class Program
  {
    static void Main()
    {
      if (true)
       {
        int i =30;
        Console.WriteLine(i);
       }
      if (true)
       {
         int i =20;
        Console.WriteLine(i);
       }
      Console.ReadKey();
    }
  }
```

واحد کار

واحد کار شماره: ۶-۱	فصل ۶ - صفحه ۱۱۳ تا ۱۳۴	مدت آموزش: ۹۰ دقیقه	روش تدریس پیشنهادی:پرسش و پاسخ- بحث گروهی
عنوان: ساختار دستور if-else		هدف کلی: آموزش دستور شرطی if-else	
اهداف رفتاری: انتظار می رود دانش آموز ۱. ساختار دستور ifرا توضیح نماید. ۲. ساختار دستور if-elseرا توضیح نماید. ۳. شرط اجرا دستور-else ifرا بیان نماید. ۴. جایگاه استفاده از دستور ifرا شرح دهد. ۵. جایگاه استفاده از دستور-else ifرا شرح دهد.		اهداف پنهان درس: آشنایی با دستور if-else،شرط دستور،شرایط صحیح اجرا دستور،شناخت عملگرهای مقایسه ای هدف ایده آل: آمادگی لازم را برای قبول مفهوم پایگاه داده را پیدا می کند.	

فرایند تدریس:

ابتدا از دانش آموزان می خواهیم که، الگوریتم برداشت پول از خودپرداز را توضیح دهند. با توجه به مراحل انجام الگوریتم با پرسش سوالات زیر ذهن دانش آموزان را متوجه نکات مهم می کنیم.

- در چه صورت دستگاه، اجازه برداشت پول را به شما می دهد؟
- اگر رمز کارت را فراموش نمایند آیا می توانند پول برداشت نمایند؟

پس از آن صورت برنامه خودپرداز را روی تخته کلاس نوشته و از دانش آموزان خواسته شود تا آن را پیاده سازی نمایند. از دانش آموزان می خواهیم مراحل زیر را انجام دهند.

۱.ابتدا یک متغیر برای دریافت رمز از کاربر تعریف شود. ۲. یک دستور شرطی برای چک نمودن رمز نوشته و در صورت صحیح بودن پیغام «رمز صحیح می باشد» را در خروجی چاپ نماید. ۳. در صورتیکه رمز را اشتباه وارد کردند پیغام مناسب چاپ نماید.

در مرحله بعد یادآوری میکنیم هر کارت بانکی یک شناسه در خود دارد که توسط آن، کارت شناسایی می شود. و در هنگام اعتبارسنجی رمز کارت وارد شده برای این شناسه بررسی می شود تا مشخص شود رمز صحیح وارد شده است یا خیر؟ از هنرجویان می خواهیم برنامه را طوری تغییر دهند که در ابتدا شناسه کارت هم وارد شد و در دستور شرط ، شناسه و رمز عبور هر دو بررسی شوند. در انتها نیز با پرسیدن این سوال که: برای ثبت و بررسی اطلاعات مقدار زیاد کارت ها چه باید کرد آنها را به سمت مفهوم پایگاه داده سوق می دهیم.

واحد کار شماره : ۶-۱	فصل ۶ - صفحه ۱۱۳ تا ۱۳۴

توصیه های اجرایی:

- مقدار عبارت منطقی را در هر مرحله از اجرا بیان کنید.
- برنامه را برای فهم بهتر Trace کنید.

ارزشیابی از انتظارات:

۱. در صورتیکه شرط دستور if درست باشد ،دستورات آن اجرا می شود. ص غ

۲. در صورتیکه شرط دستور if-else درست نباشد، دستورات قسمت if اجرا میشود. ص غ

۳. در شرط دستور if از عملگرهای ریاضی می توان استفاده نمود. ص غ

۴. دستور بعد از else در صورتی اجرا می شود که شرط قسمت اول نادرست باشد. ص غ

تکالیف و پروژه های پیشنهادی:

۱. برنامه ای بنویسید که دو عدد را خوانده، اگر علامت آنها با هم متفاوت بود پیام yes و در غیر این صورت پیام no را چاپ کند.

۲. برنامه ای بنویسید که یک نمره را دریافت کند. اگر نمره وارد شده بین ۰ تا ۲۰ باشد چاپ کند نمره مجاز است و در غیر اینصورت پیغام خطا صادر شود.

۳ . قطعه کد زیر چه کاری انجام می دهد؟

```
Input= Console.Readline();
p=int.Parse(input);
If  ((m 0== 7%) || ( m 0==5%))
Console.WriteLine(p);
```

واحد کار

واحد کار شماره:۶-۲	فصل ۶ - صفحه ۱۳۴ تا ۱۳۸	مدت آموزش:۹۰ دقیقه	روش تدریس پیشنهادی: پرسش و پاسخ - بحث گروهی– نمایش(trace برنامه)
عنوان: شماره گیری سریع در تلفن همراه (speed dialer)		هدف کلی: دستور switch	
اهداف رفتاری: انتظار می‌رود دانش آموز ۱. دستور switchرا توضیح دهد. ۲. کاربرد دستور switchرا در برنامه شرح دهد.. ۳. چگونگی مقایسه نمودن یک ورودی با چند شرط را توضیح دهد.		اهداف پنهان درس: هدف ایده آل: هنرجویان برای آرایه و لیست آماده شوند.	

فرایند تدریس:

از هنرجویان بخواهیم که امکانات شماره گیری تلفن همراه را بگویند.(یا خود هنرآموز می‌تواند مستقیم به موضوع اشاره نماید)یکی از امکانات تلفن‌های همراه speed dialer یا شماره‌گیری سریع است. شماره کسانی را در این قسمت قرار می‌دهیم که بیشترین تماس را با آنها داریم، بطور مثال مادر شماره ۱،پدر شماره۲ ،خانه شماره ۳ و خواهر شماره ۴،برادر ۵ باشد. هر زمان با پدرتان کار داشته باشید، می‌توانید شماره ۵ را بزنید ولی برای دیگران باید شماره مورد نظر را شماره‌گیری نمایید.

سپس سوالات زیر مطرح می‌شود که هنرجو به طورکامل دستور switch را درک کند.

۱- این شماره ها را تا چند شماره می‌توان ادامه داد؟

۲- نوع عددی این شماره ها از چه نوع و مقداری باید باشد؟(عدد صحیح یا اعشاری)

۳- آیا می‌توان برای تمامی شماره تلفن های خود از این روش استفاده کرد؟ چرا؟

۴- اگر شماره‌ای خارج این نفرات بود، چه اتفاقی می‌افتد؟

۵- آیا می‌توان محدوده ای از شماره ها را در نظر گرفت؟(بطور مثال بین ۱تا ۴ برای مادرتان باشد)

پس از آمادگی ذهنی هنرجویان، از آنها درخواست می‌شود برنامه ای بنویسند که با گرفتن یک شماره، چاپ کند که با چه کسی تماس گرفته خواهد شد.

واحد کار شماره : ۶-۲	فصل ۶- صفحه ۱۳۴ تا ۱۳۸

توصیه های اجرایی:

- مراقب باشید هدف کلی که دستور switch است از دست نرود.
- استفاده از دستور switch به صورت کامل شرح داده شود.
- از هنرجویان ،درخواست شود که چند مثال دیگر در این زمینه بزنند.

ارزشیابی از انتظارات:

۱. مقادیر جلوی case بصورت یک مقدار ثابت می‌باشد.	ص غ
۲. جلوی مقدار case، باید ; قرار داد.	ص غ
۳. برچسب case می‌تواند یک مقدار رشته ای باشد.	ص غ
۴. برچسب case می‌تواند یک مقدار ترکیبی رشته و عدد باشد.	ص غ
۵. در صورت برقرار نبودن همه حالات، دستورات بعد از ..... اجرا می‌شود.	
۶. برای خارج شدن از دستور switch، از دستور .....استفاده می‌شود.	

تکالیف و پروژه های پیشنهادی:

۱. برنامه ای بنویسید که طبق داده های زیر نام کانال های تلویزیونی را در خروجی (با رنگهای دلخواه) نمایش دهد و اگر شماره‌ای به جز این اعداد بود یک پیغام مناسب چاپ نماید. (۱: شبکه ملی ،۲: شبکه دو،۳ شبکه ورزش،۴:شبکه دانش، ۵: شبکه استانی)

۲. برنامه ای بنویسید که دو عدد را از ورودی گرفته و با گرفتن یکی از چهار عملگر اصلی عملیات مربوطه را انجام دهد درغیر اینصورت با یک پیغام مناسب (کاراکتر وارد شده صحیح نمی باشد) برنامه را خاتمه دهد.

۳. برنامه ای بنویسید که حالتهای یک پنکه را با دستور مناسب به صورت یک پیغام ، شبیه سازی نماید. (۰: خاموش، ۱: کند، ۲:با سرعت متوسط، ۳ : با سرعت سریع)

نمونه سوال از فصل ۶

درستی یا نادرستی هر عبارت را تعیین کنید. د برای درست و ن برای نادرست.

۱- در ifهای تو در تو هر else متعلق به نزدیکترین if است.

۲- در دستور switch انتهای دستورات default نیازی به استفاده از دستور break نیست.

۳- عبارت درون پرانتز switch از هر نوع داده ای می تواند باشد

۴- اگرعملوند اول عملگر && دارای مقدارfalse باشد، مقدار عملگر دوم بررسی نمی شود.

۵- قطعه کد ;(if (a>10 دارای خطای کامپایلری است.

در سوالات زیر گزینه صحیح را انتخاب کنید.

۶- خطای قطعه کد زیر کدام است؟

```
float mark = 17;
switch (mark)
{
 case 1:
        Console.Write(“one”);
        break;
case 2:
       Console.Write(“two”);
       break;
default:
      Console.Write(“others”);
      break;
       }
```

الف) 1 Control cannot fall through from one case label ('default:') to another

ب) 1 Invalid expression term 'case'

ج) 1 A switch expression or case label must be a bool, char, string, integral, enum, or corresponding nullable type

د) 1 } expected

۷- فلوچارت زیر نشان دهنده کدام دستور است؟

الف)for ب)while ج)switch د)if

۸- اگر Aو B با هم مخالف و مقدار C نامشخص باشد، مقدار متغیر رشته ای n چیست؟

```
bool A,B,C;
if (A=b)  if (B=C)  n="Ali";   else  n="Reza";   else n="Omid";
```

الف)Ali ب)Reza ج)Omid د)رشته تهی

۹- نتیجه اجرای قطعه کد زیر چیست؟

```
if (true); Console.Write("true"); else Console.Write("false");
```

الف)true ب)true false ج)false د)پیام خطا

در جملات زیر، جاهای خالی را با عبارات مناسب پرکنید.

۱۰- بالاترین اولویت بین عملگرهای منطقی، متعلق به عملگر ..... است.

۱۱- در عبارات ترکیبی شامل عملگرهای ریاضی و منطقی، اولویت عملگرهای ..... بیشتر است.

۱۲- در ساختار switch عبارت داخل پرانتز نمی تواند از نوع .....باشد.

ارتباط تمارین فصل و اهداف رفتاری

ردیف	اهداف رفتاری فصل ششم	خودآزمایی	تمرینات برنامه نویسی
۱.	انواع عملگرهای مقایسهای را توضیح دهد و انها را در برنامه استفاده کند.	۱و۶	
۲.	ساختار دستورif و if_else را در برنامه های شرطی خود استفاده کند.	۲و۵	۳و۱۳
۳.	مفهوم عملگرهای منطقی را شرح دهد و انواع آنها را نام ببرد.	۱	
۴.	از عملگرهای منطقی به طور صحیح در برنامه ها استفاده نماید.	۴	۳و۴
۵.	با استفاده از جدول عملکردهای منطقی، نتیجه ی عبارتهای ترکیبی منطقی و مقایسهای را بدست آورد.		۲و۱۳
۶.	اولویت عملگرها را با استفاده از جدول تقدم عملگرها در حل عبارات ترکیبی به کار بندد.		
۷.	دستور switch را در برنامه های خود به کار گیرد.		۱و۵و۸و۱۰
۸.	با ترکیب دستورهای شرطی، برنامه-هایی با ساختار پیچیده شرطی بنویسد.		۲و۴و۵و۶و ۹

بودجه بندی فصل ها

موضوع : دستورهای شرطی

هدف کلی: شناخت عملگرهای مقایسه ای و منطقی
-ساختار دستور **if-else**

ساعات تئوری: ۳ ساعات عملی: ۵

فعالیتهای آموزشی موازی

-بررسی برنامه های نمونه و کاربردی
-نحوه تبدیل یک فلوچارت به برنامه
-استفاده از **MSDN (help)** برای دیدن ساختار **if**

فعالیتهای رقابتی و ارزشیابی

- ارزشیابی ازفصل پنجم
- آزمون عملی به صورت رقابتی از مبحث شرط
- ارزشیابی به صورت سوالات چندگزینه ای و حل آن ها

فصل ششم
هفته سوم و چهارم
اسفند ماه

بودجه بندی فصل ها

فصل ششم
هفته سوم و چهارم فروردین

موضوع : دستورهای شرطی

هدف کلی: آشنایی با اولویت عملگرها در یک عبارت ترکیبی-دستور switch

ساعات تئوری: ۶ **ساعات عملی:** ۱۰

فعالیتهای آموزشی موازی

- یادآوری به دلیل وجود گپ عید نوروز
- بررسی و حل تمرین های فصل ششم
- استفاده از MSDN (help) برای دیدن ساختار switch

فعالیتهای رقابتی و ارزشیابی

- ارزشیابی فقط از Trace نمونه های شرطی
- تهیه پوستر عملگرهای منطقی، ساختار شرطی و نمونه ها به وسیله هنرجویان

آموزش فصل ۶

صفحات ۱۲۸-۱۳۰

مدرس:
خانم جمیله درساره

شهر بندرعباس
استان هرمزگان

زمستان ۱۳۹۴

فصل ۷

دستورات تکرار(حلقه‌ها)

اهداف رفتاری

کاربرد حلقه در برنامه را توضیح دهد.

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

عملکرد و کاربرد دستور حلقه while را توضیح دهد.

عملکرد و کاربرد دستور حلقه for را توضیح دهد.

برنامه های کاربردی را با حلقه تکرار بنویسد.

در برنامه های خود break و continue را در صورت لزوم به کار بندد.

ضرورت استفاده از حلقه های متداخل را توضیح دهد و آنها را در برنامه های خود به کار بندد.

دستورات تکرار(حلقه‌ها)

کلیدواژه ها: دستورات حلقه، for، while، foreach، do… while، break، continue، حلقه های متداخل

مقدمه

گاهی در برنامه ناچار به تکرار برخی عملیات هستیم. تکرار یک دستور در برنامه باعث افزایش طول کد می شود به خصوص در مواردی که تعداد این تکرار بسیار زیاد باشد. در اینگونه موارد بهتر است دستور یا دستورات یک بار نوشته شوند و ترتیبی دهیم تا برنامه آنها را به تعداد مورد نیاز تکرار کند. حلقه ها در برنامه نویسی برای تکرار تعدادی از دستورات به کار می روند.

۷-۱- ساختارهای تکرار

در بعضی از برنامه ها باید یک یا چند دستور چندین بار تکرار شود. مثلا اگر قرار باشد ۳ عدد از ورودی بگیریم می توان سه متغیر تعریف کرد و سه عدد گرفت. به شکل زیر:

```
Int32 num1, num2, num3;
Console.Write("Enter 3 numbers : ");
num1 = Int32.Parse(Console.ReadLine());
num2 = Int32.Parse(Console.ReadLine());
num3 = Int32.Parse(Console.ReadLine());
```

حال اگر بخواهیم ۱۰۰ عدد از ورودی بگیریم دیگر این روش منطقی نیست و باید از دستورات تکرار استفاده نمود.

ممکن است در تکرار دستورات، تعداد دفعات تکرار مشخص باشد یا نباشد. بنابراین دستورات تکرار (حلقه ها) به دو نوع معین و نا معین تقسیم می شوند. حلقه های معین دارای تعداد تکرار مشخص و حلقه های نامعین دارای تعداد تکرار نامشخص می باشند.

در C# چهار نوع دستور برای ایجاد حلقه وجود دارد: **for، foreach، while و do… while**

جدول ۷-۱: انواع حلقه ها در C#

الف) while(عبارت منطقی) دستور;	ب) do دستور; while(عبارت منطقی);
ج) for(; ;) دستور ;	د) foreach(in) دستور ;
دستورات تکرار در C#	

خوب است بدانیم که ماهیت حلقه های تکرار حالت خاص استفاده از دستور goto می شود. دستور goto اجرای برنامه را به محل مشخص شده ای از دستورات قبل یا بعد منتقل می کند.

زمانی که goto اجرا را به دستورات قبلی بازگرداند به نوعی یک حلقه ایجاد خواهد کرد.

نمونه ای از دستور goto در زبان C#:

```
lbl:
j++;
Console.WriteLine(j);
if (j < 7)
        goto lbl;
```

این قطعه کد مقدار j را نمایش می دهد و سپس بررسی می کند اگر مقدار j کوچکتر از ۷ باشد دوباره به محل برچسب lbl برمی گردد و دستورات بعد از آن اجرا می شوند.

استفاده از این دستور در برنامه های سطح بالا به دلیل غیراستاندارد شدن کد و کاهش خوانایی برنامه، توصیه نمی شود.

در زبان اسمبلی که یک زبان سطح پایین می باشد برای انتقال اجرا به محل خاصی از کد، از دستوراتی همچون JMP، JGE ، JNL و ..... استفاده می شود.

```
cmp     ax, bx
jge     notLess
jmp     axLess
notLess:  ....
axLess:   ....
```

۷-۱-۱- دستور تکرار شرطی while

اگر تعداد تکرار مشخص نباشد باید با استفاده از یک شرط، تعداد تکرار حلقه را کنترل کرد و بتوان آن را خاتمه داد. در دستور while می توان دستور یا دستوراتی را تکرار کرد و با استفاده از یک عبارت شرطی یا منطقی پایان اجرا را کنترل کرد.

جدول ۷-۲ : ساختار دستور while

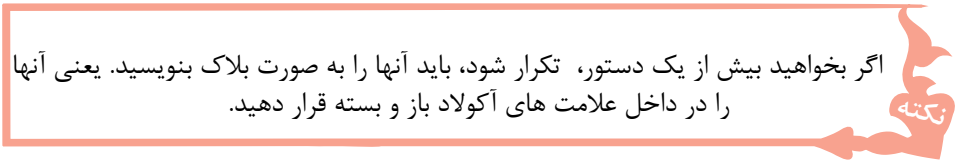
ساختار While	مثال

while (شرط)
{
بدنه
}
	while (b>0)
	{
	System.Console.
	WriteLine(b%10);
	b=b/10;
	}

به مثال زیرتوجه کنید:

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

در قطعه کد بالا اگرمقدار x کمتر از ۱۰۰ باشد دستور داخل while اجرا می شود. این برنامه اعداد ۱ تا ۹۹ را چاپ می کند.



مانند مثال زیر :

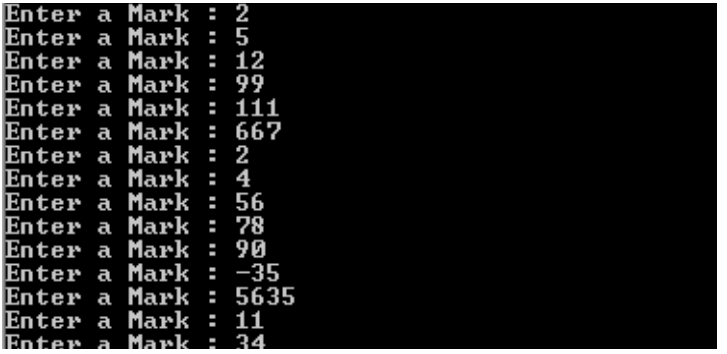
```
x = Int32.Parse(Console.ReadLine());

while (x < 100)
{
    Console.WriteLine(''x='' + x++);
     x = Int32.Parse(Console.ReadLine());
}
```

برنامه ۷-۱: برنامه ای بنویسید که تعدادی نمره از ورودی بگیرد.

این برنامه دارای دستوراتی است که باید تکرار شوند؛ بنابر این می توان از دستور while استفاده کرد. این برنامه نیاز به یک متغیر ورودی دارد . در این برنامه شرطی برای توقف دریافت نمره ها مشخص نشده است. بنابراین می‌توان شرط while را true یا یک عبارت true مانند (۲==۲) قرار داد تا دستورات داخل while تکرار شوند.

```
using System;
  class Program
  {
    static void Main()
    {
      int num;
      while (true)
      {
        Console.Write("Enter a Mark : ");
        num = Int32.Parse(Console.ReadLine());
      }
    }
  }
```

با اجرای این برنامه، چون شرط تکرار همیشه درست است ورود نمره ها متوقف نمی شود. به اصطلاح برنامه در حلقه بی پایان افتاده است.



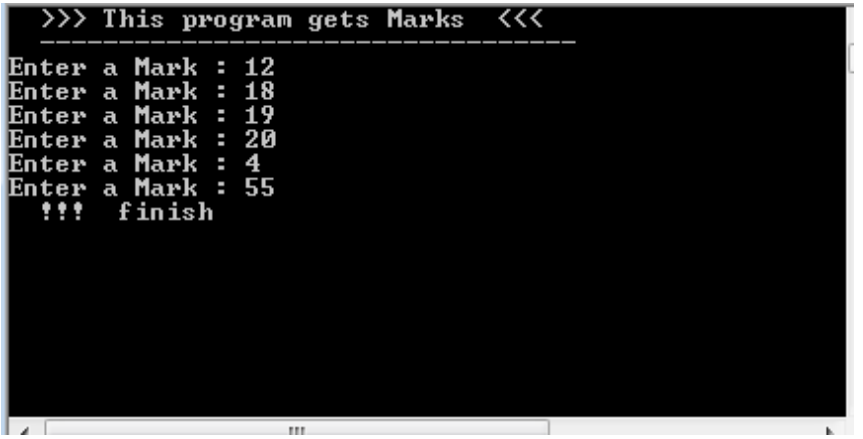
شکل ۷-۱ : خروجی برنامه۷-۱

نکته

برای خروج از حلقه پایان می توان از کلیدهای ctrl+ break یا ctrl+c استفاده نمود، می توان از شرط های ترکیبی در قسمت شرط حلقه `while` استفاده نمود.

برنامه ۷-۲: برنامه ای بنویسید که تعدادی نمره از ورودی بگیرد .
همان طور که می دانیم نمرات بین ۰ تا ۲۰ هستند پس می توان شرط ادامه تکرار را، وارد کردن نمرات بزرگتر مساوی صفر و کوچکتر مساوی ۲۰ در نظر گرفت.

```
using System;
  class Program
  {
     static void Main()
     {

        int num;
        Console.WriteLine(“  >>> This program gets Marks  <<<   “);
        Console.WriteLine(“  ---------------------------------”);
        Console.Write(“Enter a Mark : “);
        num = Int32.Parse(Console.ReadLine());
        while (num >= 0 && num <= 20 )
        {
           Console.Write(“Enter a Mark : “);
           num = Int32.Parse(Console.ReadLine());
        }
       Console.WriteLine(“  !!!  finish “);
        Console.ReadKey();
     }
  }
```



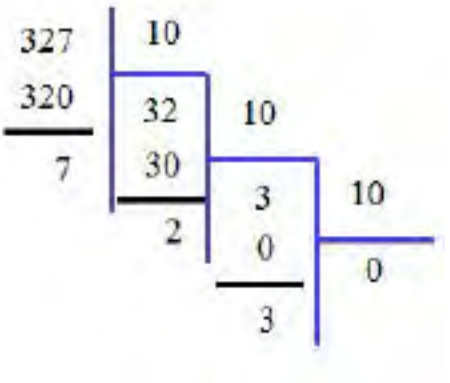
شکل ۷-۲ :خروجی برنامه ۷-۲

برنامه ۷-۳: برنامه ای بنویسید که یک عدد طبیعی از ورودی بگیرد و مقلوب آن را چاپ کند .

عدد ۳۲۷ را در نظر بگیرید: مقلوب آن عدد ۷۲۳ است.

یکی از راه حل هایی که می توان برای این برنامه در نظر گرفت این است که ارقام عدد ورودی را جدا کرده، در کنار هم چاپ نماییم.

به مراحل زیر دقت کنید :



همان طور که مشاهده می کنید با تقسیمات متوالی بر ۱۰ می توان ارقام عدد را جدا نمود (باقیمانده هر بار تقسیم). این عمل از رقم یکان صورت می گیرد و هر بار باقیمانده ها را می توان چاپ نمود. این عمل را برای خارج قسمت عدد ادامه می دهیم تا به خارج قسمت صفر برسیم.

```
using System;
class Program
{
   static void Main(string[] args)
   {
      int n,r;
      Console.Write("Enter a number : ");
      n = Int32.Parse(Console.ReadLine());
      while (n != 0)
      {
         r = n % 10;
         Console.Write(r);
         n /= 10;
      }
      Console.ReadKey();
   }
}
```

شکل۷-۳ : خروجی برنامه ۷-۳

۷-۱-۲-دستور تکرار شرطی do..while

این دستور همانند دستور while می باشد با این تفاوت که عبارت شرطی در انتهای حلقه بررسی می شود و در نتیجه حداقل یک بار دستورات حلقه اجرا می شود.

جدول۷-۳ : ساختار do-while

ساختار do..while	مثال

do
{
بدنه
}
while(شرط);
	do
	{
	ch= System.Console.ReadKey().
	KeyChar;
	str += ch;
	}
	while(ch!=' ');

برنامه ۷-۴: برنامه ای بنویسید که تعدادی نمره از ورودی بگیرد .
در برنامه ۲ مشاهده شد برای همین برنامه قبل از حلقه یک بار نمره گرفته شد.اما با استفاده از دستور do…while نیازی به دریافت نمره قبل از حلقه نیست زیرا ابتدا دستورات حلقه اجرا شده، سپس شرط بررسی می‌شود.

```
using System;
  {
    static void Main()
          {

            int num;
            Console.WriteLine(" >>> This program gets Marks <<< ");
            Console.WriteLine(" ----------------------------------");
            do
          {
            Console.Write("Enter a Mark : ");
            num = Int32.Parse(Console.ReadLine());
          } while (num >= 0 && num <= 20 );
            Console.WriteLine(" !!! finish ");
            Console.ReadKey();
          }
  }
```

۷-۱-۳- دستور تکرار for

یکی دیگر از دستورات تکرار، دستور for است. بهترین مورد استفاده از این نوع حلقه هنگامی است که نیاز به یک شمارنده داشته باشیم. ساختار کلی دستور for به صورت زیر است:

```
for( دستور انتهای حلقه ;  شرط حلقه ; مقدار اولیه = نام متغیر )
{
عملیاتی  که باید تکرار شود
}
```

سه بخش دستور for به ترتیب عبارتند از:

۱. **تعریف و مقدار دهی اولیه‌ی شمارنده:** این دستور صرف نظر از تعداد دفعات تکرار تنها یک بار در شروع حلقه اجرا می‌شود. در این قسمت می‌توان تنها یکی از حالتهای زیر را به کاربرد.
   - تعریف متغیر و مقداردهی اولیه: در این صورت متغیر محلی برای دستور for است و خارج از حلقه معتبر نیست.
   - استفاده از دستورات انتساب(عملگر=)، افزایشی(عملگر++) و کاهشی(عملگر--)، فراخوانی متدها و ایجاد شیء با استفاده از دستور new که با علامت کاما از یکدیگر جدا می‌شوند.

۲. **شرط تکرار حلقه:** که یک عبارت منطقی است و می‌تواند یک شرط یا ترکیبی از چند شرط باشد و برای تصمیم گیری درباره اجرای حلقه یا خروج از حلقه بررسی می‌شود.

۳. **دستور انتهای حلقه:** دستور یا دستوراتی که در انتهای هر تکرار باید انجام شود و می تواند شامل چند دستور از نوع انتساب(عملگر=)، افزایشی(عملگر++) و کاهشی(عملگر--)، فراخوانی متد و دستور new برای ایجاد شیء باشد که با علامت کاما از یکدیگر جدا می‌شوند.

مثال:

```
for (int i = 0;i<10; i++ )
{
        System.Console.WriteLine (i);
}
```

خروجی این برنامه چاپ اعداد ۰ تا ۹ در صفحه کنسول است.

باید توجه داشت که:

- در مواردی مانند مثال بالا متغیر i به صورت محلی در داخل دستور for تعریف شده و در خارج حلقه معتبر نیست.
- تمام عبارات این دستور اختیاری است و می توان این دستور را به صورت

```
for ( ; ; )
```

نوشت، که در این صورت یک حلقه بی پایان شکل خواهد گرفت.

- در قسمت شرط، می توانیم هر عبارت منطقی دلخواه را بنویسیم. این حلقه هربار قبل از شروع عملیات تکراری بررسی می‌شود و درصورت برقرار بودن شرط ، عملیات تکراری یک بار دیگر انجام می‌شود.

مثال: در کد

```
for (int i = 1; i++ < 10; System.Console.Write(i)) ;
```

هم چنانکه ملاحظه می شود، عملگر افزایشی در قسمت شرط آورده شده و عبارت سوم یک دستور write می‌باشد. بدین ترتیب پس از مرور شرط یک واحد به `i` اضافه گشته و در انتهای هر تکرار ،دستور write اجرا شود. نتیجه اجرای این دستور چاپ ۲۳۴۵۶۷۸۹۱۰ در کنسول است.

مثال: در کد

```
for (int i = 1; i < 5; System.Console.Write(i), i++) ;
```

در بخش سوم حلقه یعنی در قسمت دستورات انتهایی می توانیم دستورات را با کاما جدا کنیم. بنابراین ابتدا دستور `write` انجام خواهد شد و سپس عملگر افزایشی روی `i` عمل خواهد کرد. نتیجه اجرای این دستور چاپ ۱۲۳۴ در کنسول است.

مثال:در کد

```
int i = 1;

for (startAlarm();i < 5; System.Console.Write(i++)) ;
```

و با در نظر گرفتن تعریف تابع startAlarm به صورت زیر:

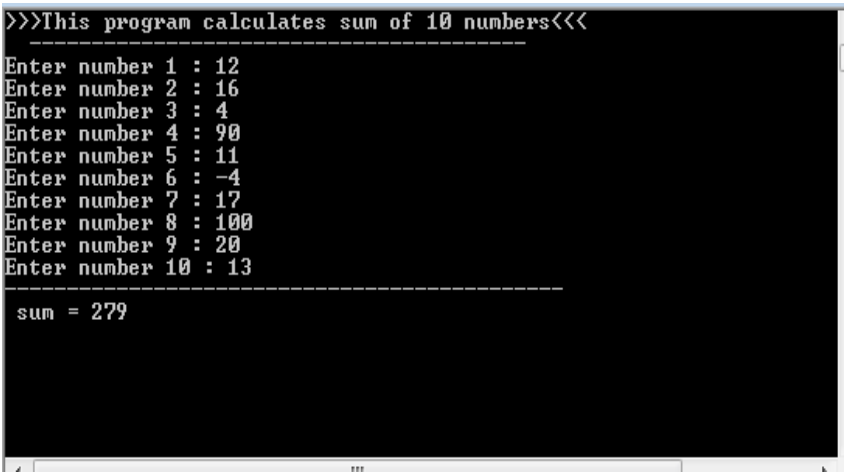
```
private static void startAlarm () { console.Write(“Started..”); }
```

خروجی Started..1234 را خواهیم داشت.زیرا المان اول دستور for ، فراخوانی تابع startAlarm می باشد و در این تابع ، متد (“Started..”)Write اجرا می شود. بعد از فراخوانی تابع، ادامه کار حلقه با بررسی شرط و انجام دستورات، صورت می پذیرد.

برنامه۷-۵: برنامه ای بنویسید که ۱۰ نمره از ورودی گرفته و مجموع آنها را محاسبه کرده و چاپ کند.

در این برنامه می توان از شمارنده حلقه for برای کنترل تعداد تکرار حلقه استفاده نمود.
در داخل دستور for هر بار یک نمره از ورودی گرفته می شود و پردازش لازم (در اینجا جمع نمرات) بر روی آن انجام می شود.

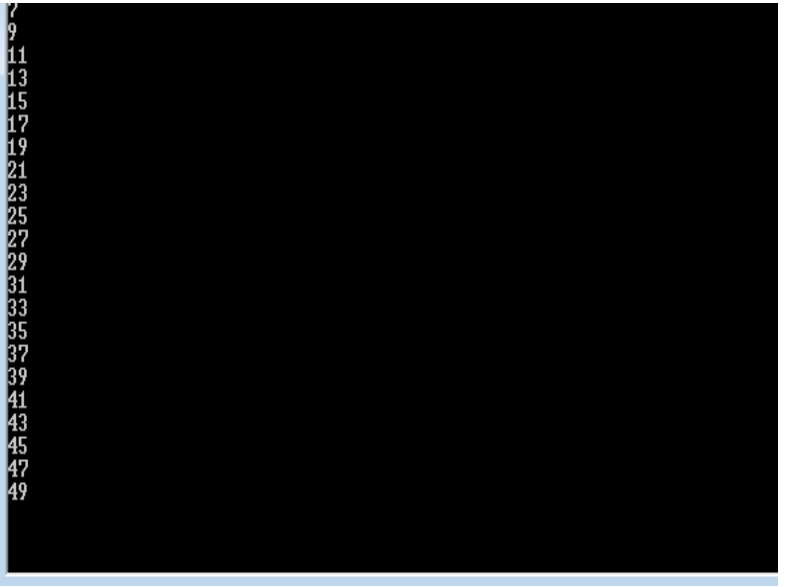
```
using System;
  class Program
  {
    static void Main(string[] args)
    {
      int num,sum = 0;
      Console.WriteLine(“>>>This program calculates sum of 10 marks<<< “);
      Console.WriteLine(“  ----------------------------------------”);

      for (int i=1 ; i <= 10; ++i)
      {
        Console.Write(“Enter number “ + i +” : “);
        num = Int32.Parse(Console.ReadLine());
        sum += num;
      }
      Console.WriteLine(“-------------------------------------------”);
      Console.WriteLine(“ sum of marks = “+ sum);
      Console.ReadKey();
    }
  }
```



شکل۷-۴ : خروجی برنامه ۷-۵

برنامه ۷-۶: برنامه ای بنویسید که اعداد فرد طبیعی کوچکتر از n را چاپ کند.
در این برنامه ابتدا باید عدد n را از ورودی بگیرد . اولین عدد فرد طبیعی از ۱ شروع می شود و هر عدد فرد بعدی ، ۲ واحد بیشتر از عدد فرد قبلی است.

```
Using System;
class Program
{
  static void Main(string[] args)
  {
          int n;
           Console.Write("Enter a number : ");
            n = Int32.Parse(Console.ReadLine());
           for (int i = 1; i <= n; i+=2)
          {
            Console.WriteLine(i);
           }
          Console.ReadKey();
  }
}
```



شکل۷-۵ : خروجی برنامه ۷-۶

برنامه۷-۷: برنامه ای بنویسید که عدد n را بگیرد و فاکتوریل آن را محاسبه کرده، چاپ کند.

تعریف فاکتوریل n (n!): حاصل ضرب اعداد ۱ تا n .

بدین منظور از فرمول ضرب استفاده می نماییم:

```
using System;
class Program
{
  static void Main(string[] args)
  {
          int num , fact ;
           Console.Write("Enter a number : ");
            num=Int32.Parse(Console.ReadLine());
            fact = 1;
           for (int i = 1; i <= num; ++i)
          {
           fact *= i;
           }
          Console.WriteLine("Factorial "+num+" ("+num+"!) is "+ fact);
    Console.ReadKey();
    }
  }
```

شکل ۷-۶ : خروجی برنامه ۷-۷

۷-۱-۴-دستور foreach

این دستور به ازای هر عضو یک مجموعه قابل شمارش، بلوک دستور را اجرا می کند. بنابراین کاربرد این حلقه هنگامی است که مجموعه ای از عناصر را داشته باشیم و بخواهیم با مرور هر عنصر، عملیاتی را انجام بدهیم.

جدول ۷-۴ : ساختار دستور foreach

ساختار foreach	مثال

foreach (تعریف متغیر in مجموعه قابل شمارش)
{
بدنه
}
	string st= “foreach loop”;
	
	foreach (char c in st)
	{
	System.Console.WriteLine(c);
	}

در دستور foreach شرط تکرار و شمارنده به صورت صریح مشخص نمی شود بلکه یک متغیر از جنس عناصر مجموعه تعریف شده، در بلوک داخلی حلقه از آن استفاده می شود. باید توجه داشت که:

- این حلقه جایگزینی برای دستور for در استفاده از اعضاء یک مجموعه می باشد بنابراین اگر این دستور و یا معادلش در زبان سطح بالایی وجود نداشته باشد به وسیله دستور for قابل جایگزینی خواهد بود. به عنوان مثال در C++ می توانیم مثال بالا را به این صورت پیاده سازی نماییم:

```
string st= “foreach loop”;
for(int i=0;i<st.Length; i++)
{
```

```
std::cout << st[i];
}
```

همان طور که ملاحظه می شود این پیاده سازی فقط مرور حروف رشته را شبیه سازی می کند. دستور for مطرح شده دارای تعریف صریح شمارنده می باشد و نیاز به بدست آوردن تعداد حروف رشته (توسط st.Length) دارد در حالی که دستور foreach این گونه نیست.

- تعریف صریح متغیر قبل از کلمه کلیدی in الزامیست و نمی توان آن را قبل از بلوک foreach تعریف نمود.
- حلقه foreach روی مجموعه های قابل شمارش اعمال می شود و یک حلقه معین محسوب می شود.

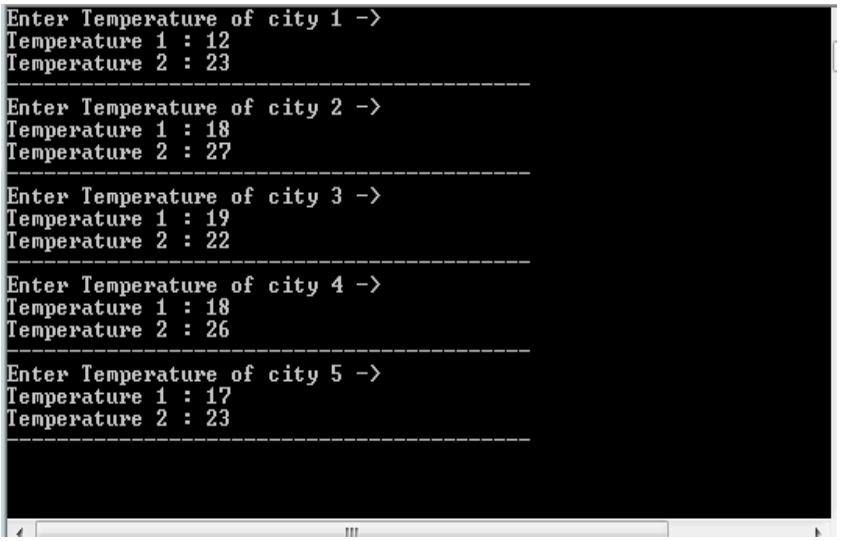
۷-۲- حلقه های تو در تو

بلوک دستورات یک حلقه می تواند دارای حلقه یا حلقه های دیگری باشد در این صورت حلقه های تودرتو یا متداخل شکل می گیرند.

برنامه ۷-۸: برنامه ای بنویسید که کمترین و بیشترین دمای ۵ شهر را از ورودی بگیرد.

در این برنامه برای ۵ شهر هر بار دو دما را از ورودی می گیریم.

```
using System;
class Program
{
  static void Main(string[] args)
  {
    int m;
    for (int i = 1; i <= 5; ++i)
    {
      Console.WriteLine("Enter Temperature of city " + i + " ->");
      for (int j = 1; j <= 2; ++j)
      {
        Console.Write("Temperature "+j+" : ");
        m = Int32.Parse(Console.ReadLine());
      }
      Console.WriteLine("----------------------------------------");
    }
      Console.ReadKey();
    }
  }
```



شکل ۷-۷ : خروجی برنامه ۷-۸

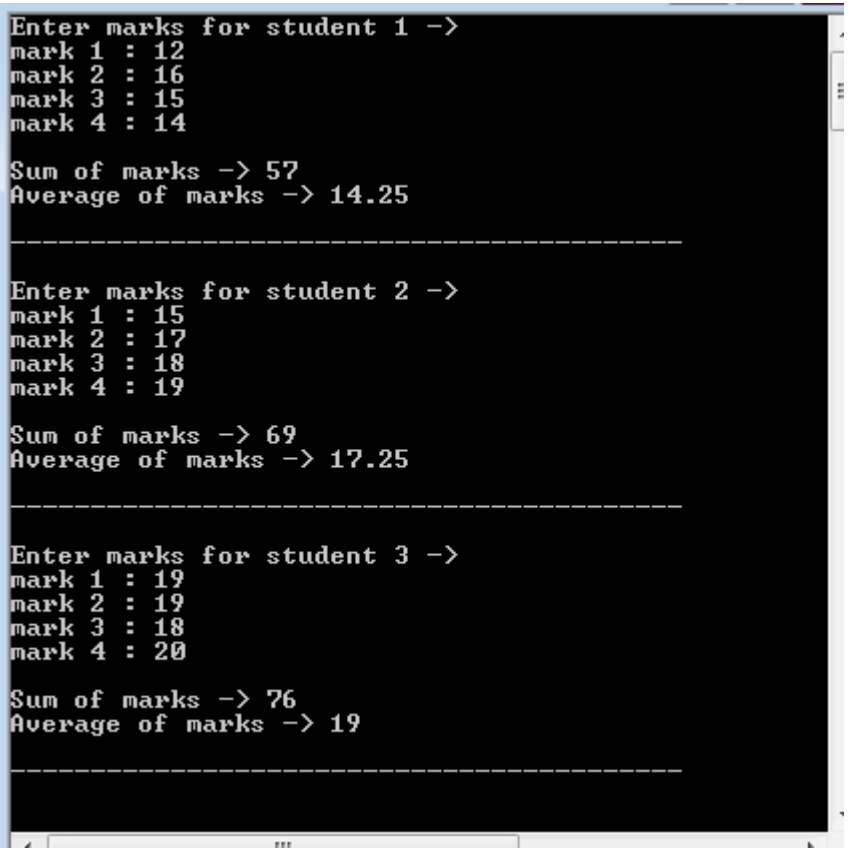
برنامه ۷-۹ : برنامه ای بنویسید که مجموع و میانگین ۴ نمره ۳ دانش آموز را محاسبه نماید .

معمولا در برنامه هایی که با دو for حل می شود، for اول مربوط به داده های اصلی برنامه است. در این برنامه for اول برای کنترل تعداد دانش آموزان و for دوم برای گرفتن ۴ نمره هر دانش آموز بکار می رود.

```
using System;
class Program
{
  static void Main(string[] args)
  {
      int num , sum ;
    for (int i = 1; i <= 3 ; ++i)
    {
      Console.WriteLine("Enter marks for student " + i + " ->");
      sum = 0;
       for (int j = 1; j <= 4; ++j)
      {
        Console.Write("mark "+j+" : ");
        num = Int32.Parse(Console.ReadLine());
        sum += num;
      }
      Console.WriteLine("\nSum of marks -> " +sum);
```

```
        Console.WriteLine("Average of marks -> " + sum/4.0);
        Console.WriteLine("\n----------------------------------------\n");
    }
        Console.ReadKey();
    }
}
```

با استفاده از عبارت کنترلی \n در عبارت توضیحی دستور Write می توان اشاره گر را در صفحه خروجی به سطر بعدی منتقل کرد.



شکل ۷-۸ : خروجی برنامه ۷-۹

برنامه۷-۱۰: برنامه ای بنویسید که جدول ضرب ۱۰ در ۱۰ را چاپ کند.

```
using System;
class Program
{
   static void Main(string[] args)
           {
           for (int i = 1; i <= 10; ++i)
               {
               for (int j = 1; j <= 10; ++j)
                  Console.Write(i * j+"\t");
              Console.WriteLine();
       }
       Console.ReadKey();
       }
   }
```

```
1       2       3       4       5       6       7       8       9       10
2       4       6       8       10      12      14      16      18      20
3       6       9       12      15      18      21      24      27      30
4       8       12      16      20      24      28      32      36      40
5       10      15      20      25      30      35      40      45      50
6       12      18      24      30      36      42      48      54      60
7       14      21      28      35      42      49      56      63      70
8       16      24      32      40      48      56      64      72      80
9       18      27      36      45      54      63      72      81      90
10      20      30      40      50      60      70      80      90      100
```

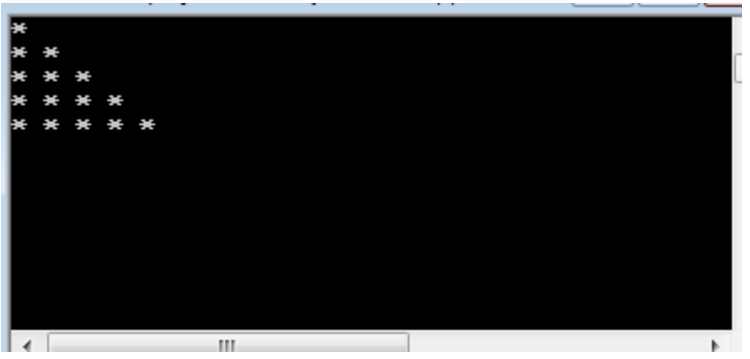
شکل۷-۹ : خروجی برنامه ۷-۱۰

در این برنامه از دو for برای چاپ خروجی استفاده شده است. for اول برای کنترل خروجی و تعداد سطرها استفاده می شود و for دوم برای چاپ مقادیر هر سطر بکار می رود. در هر سطر مقدار متغیر j چاپ می شود.

عبارت \t در دستور Write باعث ایجاد یک پرش (tab) در خروجی می شود.

برنامه۷-۱۱: برنامه زیر را در VS نوشته و خروجی آن را بررسی نمایید.

```
using System;
class Program
{
  static void Main(string[] args)
        {
          for (int i = 1; i <= 5; ++i)
            {
              for (int j = 1; j <= i; ++j)
                Console.Write("* ");
              Console.WriteLine();
            }
          Console.ReadKey();
        }
  }
```



شکل۷-۱۰ : خروجی برنامه ۷-۱۱

در این برنامه در for دوم تعداد دفعات تکرار بستگی به مقدار `i` دارد.

۷-۳-نکاتی دیگر درباره حلقه ها

- اگر پس از دستور حلقه سمی کولن (;) گذاشته شود دارای دستورات داخلی نخواهد بود .در این حالت خصوصا در مورد حلقه while امکان تکرار بی نهایت خواهد بود:

```
b=1;
while (b>0);
```

- حلقه های for و while در زبان C# قابل جایگزینی به همدیگر می باشند در حالی که در زبانی مانند VB تنها حلقه for قابل پیاده سازی با while است.
- به طور کلی نوع while نامعین و نوع for معین ذکر می شود. اما در واقع با توجه به کاربرد ممکن است این قضیه بر عکس شود:

```
for (;b>0;)
{
        System.Console.WriteLine(b%10);
        b=b/10;
}
------
int i = 0;
while (i<10)
{
         System.Console.WriteLine (i);
         i++;
}
```

- در تمام حلقه های ذکر شده، دو دستور break و continue می تواند بکار رود.break برای توقف حلقه و بیرون رفتن از آن استفاده می شود. هرگاه بخواهیم اجرای دستورات بلوک حلقه متوقف شده و اجرا به محل بررسی شرط منتقل شود از دستور continue استفاده می شود.

مثال: قطعه کد زیر اعداد ۹ و ۱۰ را نمایش می دهد:

```
for (int i = 1; i <= 10; i++)
{
        if (i < 9)
        {
                continue;
        }
        Console.WriteLine(i);
}
```

در این کد تا زمانی که i<۹ می باشد با اجرای continue ، حلقه دستور i++ را اجرا کرده، به ابتدای حلقه باز می گردد.

مثال:

```
for (int i = 1; i <= 100; i++)
{
        if (i == 5)
        {
                break;
        }
        Console.WriteLine(i);
 }
```

در این کد اگر i برابر با ۵ باشد دستور break اجرا می شود و در نتیجه تکرار متوقف خواهد شد و اجرا به دستور بعد از حلقه منتقل می شود.

- ممکن است حلقه به صورت برنامه ریزی شده یا خطای منطقی به صورت نامتناهی تکرار شود .در محیط کنسول اگر بخواهیم برنامه را متوقف کنیم از کلید های ترکیبی CTRL+BREAK یا CTRL+C استفاده می کنیم.توجه شود که این کلید ها موجب بروز رویداد Console.CancelKeyPress می شود و در حالت عادی کل برنامه را متوقف می سازد.

۷-۴- حلقه ها در زبان های دیگر

حلقه ها ی while و for با مشابهت بسیار در اکثر زبان های سطح بالا وجود دارند:

c,c++,D,c#,java,javascript,...

```
while (c > 1)
  {
    c=c-1;
  }
```

pascal

```
while c > 0 do
  begin
      Counter := Counter - 1
  end;
```

c,c++,D,c#,java,javascript,...

```
while (c > 1)
  {
    c=c-1;
  }
```

pascal

```
while c > 0 do
  begin
      Counter := Counter - 1
  end;
```

c,c++,D,c#,java,javascript,...

```
for (i=1;i<5;i++)
  {
    …
  }
```

Basic

```
For I = 1 to 5;
…
Next I
```

شکل ۷-۱۱ : نمونه ای از کاربرد حلقه while در زبان های مختلف

Fortran

```
do counter = 1, 5, 1
  …
end do
```

Matlab

```
for i = 1:5
  …
end
```

شکل ۷-۱۲ : نمونه ای از کاربرد حلقه **for** در زبان های مختلف

اما در مورد دستور foreach موضوع کمی متفاوت تر می باشد.گرچه دستور foreach در معدودی از زبان ها دیده می شود اما مانند دستورهای while و for عمومیت ندارد. برخی از زبان های سطح بالا معادل این حلقه را دارند و برخی دیگر مانند C++ چنین حلقه ای را در دایره دستورات خود جای نداده اند.

Delphi

```
for enumerator in collection do
begin
 …
end;
```

Java

```
for (type item: iterableCollection) {
  …
}
```

Javascript

```
for (var key in object) {
//کار بر روی object[key] خواهد بود
}
```

Matlab

```
for item = array
…
end
```

شکل۷-۱۳ : نمونه ای از دستورهای معادل حلقه **foreach** در زبان های دیگر

واژگان

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	**Boolean expresion**	عبارت منطقی
۲	**Conditional command**	دستور شرطی
۳	**Compile error**	خطای کامپایل
٤	**Deciding**	تصمیم گیری

پرسش و پاسخ

۱- با توجه به قانون حوزه اعلان متغیر، آیا در قطعه کد زیر، خطایی رخ می دهد؟

```
for ( int x = 0; x<5 ; x++)
{
Console.WriteLine(“MPC”);
}
x=25;
```

بله خطا رخ میدهد.

حوزه کاری x با پایان یافتن حلقه ی for تمام میشود و نمی توان بیرون از حلقه ی for از متغیر x استفاده کرد.

۲- با توجه به قانون حوزه اعلان متغیر، علت خطا در قطعه کد زیر چیست؟

```
for ( int x = 0; x<5 ; x++)
{
Consoles.WriteLine(“MPC”);
}
int x;
```

حوزه کاری x با پایان یافتن حلقه ی for تمام می شود و نمی توان بیرون از حلقه ی for از متغیر x استفاده کرد. ولی بیرون از حلقه هم نمی توان دوباره آن متغیر را تعریف کرد.

پس در حالی که حوزه کاری هر دو x کاملا از هم جداست ولی C# از اعلان مجدد متغیر x جلوگیری می کند.

واحد کار

واحد کار شماره: ۷-۱	فصل ۷ - صفحه ۱۴۴ تا ۱۵۰	مدت آموزش: ۹۰ دقیقه	روش تدریس پیشنهادی: پیش سازمان دهنده
عنوان: حلقه While		هدف کلی: آموزش حلقه while	
اهداف رفتاری: انتظار می رود دانش آموز ۱. ساختار اصلی حلقه را شرح بدهد. ۲. شرط حلقه را درست انتخاب کند. ۳. ساختار do..while و while را درست بکار ببرد ۴. بتواند حلقه را trace نماید.		اهداف پنهان درس:عادت دادن دانش آموز به نوشتن ساختار اصلی و سپس توسعه While در حلقه های پیچیده هدف ایده آل: آشنایی با نحوه کار برخی وقفه های نرم افزاری که می توان از طریق while پیاد سازی کرد.	

فرایند تدریس:

از دانش آموزان می خواهیم که یک بانک را در نظر بگیرند که قرار است که به پدرشان کمک کرده و پولی را به حسابش واریز کنند. برای اینکار باید در صف انتظار بمانند. به هر مشتری یک شماره تعلق می گیرد.و هر مشتری بایدمنتظر بماند تا شماره اش خوانده شود.از آنها می خواهیم پرسش های مطرح شده را برای این سیستم پاسخ بدهند.

۱. چگونه شماره دهی انجام می شود؟
۲. تکراری در انجام کار می بینید؟
۳. اگر مشتری بیرون بانک کاری داشته باشد برود، شماره اش چه می شود؟
۴. آیا یک مشتری می تواند با شماره روز قبل خود وارد صف مشتری های امروز شود؟
۵. آیا بعد از پایان یافتن ساعت کار بانک ،شماره مشتری که در ساعت کاری بانک شماره گرفته و بیرون رفته ،اعتبار دارد؟
۶. آیا بانک بعد از ساعت کاری بانک ؛مشتری جدید قبول می کند؟

واحد کار شماره: ۷-۱	فصل ۷ - صفحه ۱۴۴تا ۱۵۰

توصیه های اجرایی:

- هدف اینست که در ابتدا ساختار شرط با کمترین دستور داخلی نوشته شود و کم کم حلقه توسعه پیدا کند
- بهتر است یک بار به صورت Do...While حلقه اجرا شود و نادرستی استفاده از این نحوه کاربرد حلقه، به صورت عملی نمایش داده شود.
- حتما قبل از استفاده متغیر در جلوی دستور while،آن تعریف و مقداردهی شود.
- در حلقه شمارنده تعیین شود.
- شبیه سازی حلقه for با while انجام شود

ارزشیابی از انتظارات: (سوالات ۱ تا ۸ سوال ص غ هستند)

۱. اگر شرط حلقه while درست باشد دستورات درون آن اجرا می شوند.
۲. حلقه while دارای یک شمارنده است که تعداد دور را نگهداری می کند.
۳. دستورات درون While محدودیتی از لحاظ تعداد و نوع ندارد.
۴. با trace حلقه می توانیم وضعیت اجرای حلقه را در هر دور، پیگیری نماییم.
۵. قبل از استفاده از متغیر درون حلقه ،باید آن را تعریف و مقدار دهی کرد.
۶. شرط while درون {} نوشته می شود.
۷. احتمال دارد دستورات درون حلقه While اجرا نشوند.
۸. اگر شرط حلقه خالی باشد بی نهایت بار اجرا می شود.
۹. آیا می توان از متغیری خارج از برنامه در دستور while استفاده نماییم؟

تکالیف و پروژه های پیشنهادی:

۱. تعداد مشتریانی که در یک روز کاری به بانک مراجعه کرده اند را بدست اورده و نمایش دهید .شرط خروج حلقه پایان ساعت کاری بانک است.
۲. در یک مسابقه ۲۰ سوالی رادیوئی ،شرکت کننده در این مسابقه ،تنها ۲۰ حدس برای یافتن جواب مسابقه می تواند بزند و در صورت حدس زدن جواب جایزه مسابقه را ببرد.برنامه ای برای این بخش مسابقه رادیو بنویسید.

واحد کار

واحد کار شماره: ۷-۲	فصل ۷ - صفحه ۱۵۶ تا ۱۵۸	مدت آموزش: ۹۰ دقیقه	روش تدریس پیشنهادی: مبتنی بر فکر
عنوان: حلقه for		هدف کلی: آموزش حلقه ی تکرار for	
اهداف رفتاری: انتظار می رود دانش آموز ۱. الگوریتم برنامه را به زبان ساده شرح دهد. ۲. مقدار اولیه، شرط خاتمه و گام حلقه را بتواند به درستی مقداردهی نماید. ۳. طول حلقه‌ی تکرار را به درستی محاسبه نماید. ۴. بتواند برنامه نوشته شده را با جدول صحت خط به خط اجرا نماید.		اهداف پنهان درس:آشنایی با اجزای حلقه، مقدار اولیه، گام حلقه، شرط خاتمه و دستورات حلقه هدف ایده آل:	

فرایند تدریس:

به دانش آموزان میگوییم فرض کنید دکتر برای سرماخوردگی شما دو قرص تجویز کرده که هر هشت و دوازده ساعت مصرف کنید. دکتر از هر قرص سی عدد تجویز کرده است. اگر امروز اولین روز باشد، آخرین قرص را چه روزی میل خواهید نمود.کدام قرص سریع‌تر تمام خواهد شد و به چه دلیل ؟. سپس از دانش‌آموزان می‌خواهیم با استفاده از حلقه‌ی For برنامه ای بنویسند که برای یک روز، هر هشت و دوازده ساعت قرص اول و دوم را با دستور چاپ یادآوری نماید. (بهتر است ابتدا برای هر یک از حالات یک برنامه بنویسند و سپس دو برنامه را با استفاده از یک حلقه‌ی تکرار ادغام نمایند.)

قرص هشت ساعتی: مقدار اولیه: ۰ شرط خاتمه: ۲۴ گام حلقه: ۸ دستور حلقه: چاپ (خوردن قرص هشت ساعتی)

قرص دوازده ساعتی: مقدار اولیه: ۰ شرط خاتمه: ۲۴ گام حلقه: ۱۲ دستور حلقه: چاپ (خوردن قرص دوازده ساعتی)

ترکیب دو حالت: برای این حالت، ضمن توضیح اهمیت انتخاب گام حلقه‌ی مناسب، بزرگترین مقسوم علیه مشترک دو گام حلقه‌ی حالات قبل را به عنوان گام حلقه‌ی جدید انتخاب می‌کنیم: ب.م.م(۸ و ۱۲)=۴ . مقدار اولیه و شرط خاتمه مثل حلقه های قبل است

دستورات حلقه: با توجه به بخش پذیر بودن شمارنده حلقه بر هشت یا دوازده دستور چاپ مناسب استفاده می شود

واحد کار شماره: ۷-۲	فصل ۷ - صفحه ۱۵۶ تا ۱۵۸

توصیه های اجرایی:

- حتما در داخل بلاک حلقه از متد ReadKey استفاده شود تا دانش آموز در مرحله‌ی اجرا با ایجاد مکث، تکرار را درک نماید.
- از دانش آموزان بخواهید پس از کدنویسی حتماً دستورات را خط به خط اجرا نموده و مقادیر متغیرها را در جدول صحت یادداشت نمایند.
- حلقه‌ی بی‌نهایت را نیز برای دانش‌آموزان توضیح داده و شرایط ایجاد آن را مورد بحث قرار دهید.
- در رابطه با گام حلقه‌ی افزایشی و کاهشی و ارتباط آن با مقادیر اولیه و شرط خاتمه نیز مطالبی را گوشزد نمایید. به جهت سهولت می‌توانید روش‌های متفاوت حل مسئله را در فرآیند تدریس ارائه نمایید.

ارزشیابی از انتظارات: (سوالات ۱ تا ۸ سوال ص غ هستند)

۱.شرط خاتمه در حلقه‌ی تکرار همواره باید عددی بزرگتر از مقدار اولیه باشد.
۲. شمارنده حلقه در هر بار تکرار لزوماً دارای مقداری بیشتر از تکرار قبل خواهد بود.
۳. طول حلقه‌ی تکرار را در مثال زیر بیابید.

تکالیف و پروژه های پیشنهادی:

۱. حلقه‌های تکرار زیر چه عملی را انجام می‌دهند؟(m و S متغیر هستند)

```
S=0;
For (مقدار اولیه ; شرط خاتمه ;گام حلقه)
    S=S+1;
```

```
S=0;
For (مقدار اولیه ; شرط خاتمه ;گام حلقه)
    S=S+m;
```

```
S=1;
For (مقدار اولیه ; شرط خاتمه ;گام حلقه)
    S=S*m;
```

۲.برای هر یک از حلقه های فوق با مقدار اولیه۱، شرط خاتمه ۱۰، گام حلقه ۲و m=3 ، جدول trace رسم کنید.
۳. برنامه‌ای بنویسید که به فرد بیمار هر یک ساعت اعلان کند که چند ساعت دیگر تا تناول هر قرص مانده است.

واحدکار

واحد کار شماره: ۷-۳	فصل ۷ - صفحه ۱۶۳تا ۱۶۵	مدت آموزش: ۹۰ دقیقه	روش تدریس پیشنهادی: پیش سازمان دهنده- مبتنی بر بیان فکر
عنوان: حلقه for	هدف کلی: آموزش حلقه ی تکرار for		
اهداف رفتاری: انتظار می رود دانش آموز ۱. الگوریتم مسئله را به درستی بنویسد. ۲. نوع حلقه‌ی مناسب برای هر برنامه را انتخاب نماید. ۳. با تغییر برنامه، خروجی آنرا به درستی محاسبه نماید. ۴. برنامه نوشته شده را خط به خط اجرا نماید.	اهداف پنهان درس: تبدیل مسئله به برنامه و کدنویسی آن، انتخاب و به کارگیری دستور حلقه مناسب (for یا while) برای مسائل، درک وجود مقایسه هدف ایده آل: درک نقاط بیشینه نسبی و بیشینه مطلق در مسائل و تلاش برای جامع نگری در زندگی برای یافتن نقاط بیشینه مطلق		

فرایند تدریس:

ابتدا برای دانش‌آموزان توضیح می‌دهیم که می‌خواهیم برنامه یافتن بزرگترین عدد را بنویسیم. سپس به صورت تصادفی پنج نفر از دانش‌آموزان را به بیرون از کلاس هدایت می‌کنیم. از دانش‌آموزان درون کلاس می‌خواهیم که با یک بار بررسی هر فرد، قد بلندترین شخص را مشخص نمایند و در کلاس یک جایگاه برای بلندترین فرد انتخاب می‌کنیم. با ورود نفر اول از دانش‌آموزان می‌خواهیم که فرد وارد شده را مورد ارزیابی قرار دهند. پس از کمی مباحثه، مشترکاً به این نتیجه خواهیم رسید که فعلاً فرد اول در جایگاه قرار بگیرد. با ورود فرد دوم از دانش‌آموزان انتظار می‌رود با مقایسه‌ای ساده بتوانند تعیین کنند که چه کسی می‌تواند در جایگاه قرار بگیرد. به همین ترتیب با ورود نفرات دیگر در هر مرحله از دانش‌آموزان می‌خواهیم مقایسه را انجام دهند. سپس با پرسش چند سوال به آماده‌سازی دانش‌آموزان برای کدنویسی کمک می‌کنیم: در هر لحظه چه کسی بر روی جایگاه قرار خواهد گرفت؟ دستور مقایسه چند بار تکرار خواهد شد؟ برنامه چه زمانی متوقف می‌شود؟ آیا لزومی دارد که اندازه‌ی قد تمام افراد به خاطر سپرده شود؟

سپس از دانش‌آموزان می‌خواهیم با تحلیل برنامه مثال فوق، اجزاء حلقه‌ی تکرار و دستورات درون حلقه را مشخص نمایند و برنامه‌ی مثال فوق را بنویسند.

واحد کار شماره: ۷-۳	فصل ۷ - صفحه ۱۶۳ تا ۱۶۵

توصیه های اجرایی:

- دلایل استفاده از حلقه‌ی تکرار برای دانش آموزان گوشزد شود.
- در ارتباط با نوع حلقه‌ی تکرار مورد استفاده با دانش‌آموزان بحث کنید.
- از اهمیت اجرای خط به خط و کاربرد جدول صحت غافل نشوید و پس از نوشتن برنامه، ابتدا آن را با جدول صحت اجرا نمایید.
- پس از نوشتن برنامه با تغییر اجزای حلقه از دانش‌آموزان بخواهید، خروجی را حدس بزنند.

ارزشیابی از انتظارات:

۱. آیا در این برنامه طول حلقه‌ی تکرار ثابت است؟
۲. اگر بیشترین عدد همان ابتدای اجرا وارد شود، اجرای برنامه به چه صورت ادامه می‌یابد؟
۳. اگر تعداد افراد از پنج نفر به دوازده نفر تغییر کند، در کدنویسی و اجرا چه بخشی تغییر خواهد نمود؟
۴. اگر گام حلقه به جای افزایش یا کاهش یک واحد، دارای افزایش یا کاهش دو واحد باشد و طول حلقه‌ی تکرار به تعداد افراد باقی بماند، چه رخ خواهد داد؟

تکالیف و پروژه های پیشنهادی:

۱. برنامه را طوری تغییر دهید که نام افراد هم به عنوان ورودی گرفته شود و در انتها نام قد بلندترین فرد را نیز نمایش دهد.
۲. برنامه ای بنویسید که رکوردهای زمان رسیدن به خط پایان برای شرکت کنندگان در یک مسابقه دومیدانی را وارد نموده و بهترین زمان را نمایش دهد

واحد کار

واحد کار شماره: ۷-۳	فصل ۷ - صفحه ۱۵۳تا ۱۵۶	مدت آموزش: ۹۰ دقیقه	روش تدریس پیشنهادی:نمایشی(بازی)-پرسش و پاسخ -پیش سازمان دهنده
عنوان: برنامه حدس عدد		هدف کلی: تمرین حلقه do...while	
هداف رفتاری: انتظار می رود دانش آموز ۱. الگوریتم برنامه را به زبان ساده شرح دهد و ۲. ورودی ها و خروجی های برنامه را تعیین کند. ۳. حلقه مورد نیاز این برنامه و شرط حلقه را تشخیص دهد. ۴. بتواند برنامه نوشته شده را Trace کند.		اهداف پنهان درس: تاکید روی نکاتی مانند شرط توقف حلقه، تکرار با تعداد نامشخص، شرط های درون بازی برای کمک به حدس نزدیکتر مقایسه هدف ایده آل: اشاره به روش جستجوی دودویی بدون ذکر نام و با مهارت بازی به طوریکه در بازی بعدی دو شرکت کننده با روش دودویی سریعتر به پاسخ برسند.	

فرایند تدریس:

ضمن ارتباط با مباحث قبلی، به دانش آموزان بگویید امروز با هم می خواهیم مانند یک Game Programmer فکر کنیم و برنامه یک بازی را بنویسیم. دونفر از دانش آموزان را به شیوه تصادفی انتخاب می کنیم. روش بازی را برای کل کلاس توضیح می دهیم. با راهنمایی درِگوشیه مربی، عدد دو رقمی را یک دانش آموز به ذهن می سپارد و بازی را نفر دیگر شروع می کند و با هدایت مربی اجازه می دهیم بازی تمام شود. در بین بازی اگر لازم است توضیحاتی ارایه دهید تا کلاس به سمت هدف شما هدایت شود نه هدف بازی. این سوالات پس از پایان بازی راه گشاست. حتی می توانید آنها را روی تابلو بنویسید و در برنامه از آنها استفاده کنید:

چرا بازی تمام شد؟(شرط پایانی حلقه) چرا عدد دوم حدس نزدیکتری به عدد اصلی بود؟(شرط موردنظر) چه چیزی باعث می شد عددها به عدد اصلی نزدیکتر شوند؟(شرطهای بزرگتر و کوچکتر) تا کی حدس عددها ادامه دارد؟(تکرار در حلقه)

در بازی دونفره بعدی بپرسید: چگونه می توان عددها را طوری حدس زد که در کمترین زمان ممکن به پاسخ دست یافت؟(هدایت پاسخ ها برای استفاده از تکنیک جستجوی دودویی)

نکته: ما نمی خواهیم برنامه را بنویسیم. اما می توانیم در روش حدس از آن کمک بگیریم. درست مانند اینکه در جستجوی شماره پلاک یک خانه از آن استفاده می کنیم. یک طرف خیابان پلاک های زوج و طرف دیگر فرد است و ترتیب پلاک ها در یافتن شماره مورد نظر راهگشاست.

واحد کار شماره: ۷-۴	فصل ۷ - صفحه ۱۵۳ تا ۱۵۶

توصیه های اجرایی:

- در این واحد کار حلقه تدریس نمی شود بلکه با یک مثال عینی تمرین می شود.
- هرچه مثال هایی که می آورید به زندگی روزمره نزدیک تر و عینی تر باشند، یادگیری بهتر خواهد بود.
- از دانش آموزان بخواهید مثال های عینی در زندگی روزمره از تکرار نامعین و خروج با شرط بزنند.
- چون تکنیک دودویی شیوه جالبی است و دانش آموز را علاقمند می کند، دقت کنید درگیر جستجوی دودویی نشوید.
- اهمیت Trace را برشمارید و با گذاشتن شرط اشتباه و Trace نشان دهید اجرای گام به گام می تواند اشکال زدایی انجام دهد.

ارزشیابی از انتظارات:

۱. اگر در اولین حدس، عدد درست باشد، حلقه while ... do اصلا تکرار نمی شود.
۲. اگر شرط while غلط باشد دستورات داخل آن تکرارمی شوند.
۳. حلقه while به تعداد معینی دستورات را تکرار می کند و انجام می دهد.
۴. همیشه برنامه ابتدا دستور داخل حلقه را که از do شروع می شود اجرا می کند.
۵. در تمرین «حدس عدد» می توان تعداد حدس ها را محدود و کم نمود.
۶. تکرار حدس اعداد و رسیدن به پاسخ صحیح را با استفاده از حلقه ی for هم می توان نوشت.
۷. شرط در جلوی کلمه while نوشته می شود.
۸. یک حلقه while ... do می تواند بی نهایت بار اجرا شود.
۹. اگر شرط حلقه نادرست باشد، دستور بعد از حلقه اجرا می شود.
۱۰. دستورات درون while ... do حداقل یک بار انجام می شوند.

تکالیف و پروژه های پیشنهادی:

۱. در یک فروشگاه زنجیره ای برای جمع خرید بالاتر از یک میلیون ریال ۱۰درصد تخفیف درنظر گرفته می شود. قیمت اجناس خریداری شده توسط یک مشتری را وارد کنید و پس از کسر تخفیف، مبلغ قابل پرداخت را محاسه نمایید.(شرط خروج قیمت صفر می باشد)
۲. کد بازی برنامه را به گونه ای تغییر دهید که بازی بین سه نفر انجام شود و دو نفر باهم حدس بزنند، برنده کسی است که حدسِ درست بزند.(نوسعه شرط با استفاده از عملگر | |)

نمونه سوال از فصل ۷

درستی یا نادرستی هر عبارت را تعیین کنید. د برای درست و ن برای نادرست.

۱-در دستور while، گذاشتن علامت ; بعد از پرانتز، تاثیری در اجرای حلقه while ندارد.

۲- در حلقه do-while اگر شرط برقرار نباشد، حداقل یک بار دستورات حلقه انجام می شود.

۳-دستور for(; ;); بدون ورودی خطای کامپایل دارد.

۴- در حلقه های متداخل ،باید بلوک حلقه داخلی را بعد از بلوک حلقه خارجی بست.

در سوالات زیر گزینه صحیح را انتخاب کنید.

۵- خروجی تکه برنامه زیر کدام گزینه است؟

```
a = int.Parse(Console.ReadLine());
while (a>0)
{
   b=a%10;
   if (b%2==0) d++;
   a/=10;
}
Console.WriteLine(d);
```

الف)چاپ تعداد ارقام زوج عدد a　　ب) چاپ ارقام زوج عدد a

ج) چاپ یک در میان ارقام عدد a　　د) چاپ تعداد ارقام فرد عدد a

۶- در کدام عبارت اگر مقدار متغیر Flag مساوی false باشد، حلقه ادامه پیدا می کند؟

الف) while (!Flag)　　ب) while(Flag)　　ج) while(Flag = flase)

۷- برای دستور زیر کدام درست است؟

```
for(a;b;c)
{
  بدنه حلقه
}
```

الف)دستور c را می‌توان در بدنه حلقه قرار داد.

ب)دستور a را می توان در ابتدای بدنه حلقه قرار داد.

ج)دستور c را می توان قبل از دستور for قرار داد.

د)دستور b را می توان در ابتدای بدنه حلقه قرار داد.

۸- خروجی برنامه زیر کدام است؟

```
int i;
for (i = 0; i < 5; i++) ;
    Console.Write(i);
```

الف)0 1 2 3 4 5 ب)0 1 2 3 4 ج)5 د)خطا

۹- اگر خروجی این قطعه کد به صورت زیر باشد.به جای علامت سوال درحلقه j کدام یک را می توان قرار داد؟

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

{

  for (int j = 1; j <= ?; j++)

     Console.Write("*");
  Console.WriteLine();
}
```

الف)6 ب)i ج)6-i د)5

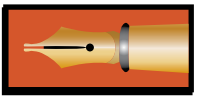
در جملات زیر، جاهای خالی را با عبارات مناسب پرکنید.

۱۰- دستورات حلقه While در صورتی تکرار می شود که عبارت منطقی حلقه..... باشد.

۱۱- در حلقه for نزولی تغییر مقدار شمارنده حلقه باید ..... باشد.

۱۲- برای جلوگیری از انجام یکی از تکرارهای حلقه می توان از دستور ..... استفاده کرد.

۱۳- برای خروج از حلقه قبل از پایان آن ار دستور ..... استفاده می شود.



جدول مشخصات آزمون فصول ۵-۶-۷

جدول این بخش به دلیل پیچیدگی در قالب **docx** از طریق وبگاه **tvoccd.sch.ir** در اختیار هنرآموزان عزیز قرار خواهد گرفت.

نمونه آزمون مبتنی بر جدول مشخصات

درستی یا نادرستی هر عبارت را تعیین کنید. **د** برای درست و **ن** برای نادرست.
(هر کدام ۰,۲۵ نمره)

۱- علامت هایی که بیانگر انجام یک عمل بر روی اعداد و داده ها هستند عملوند می گویند.

۲- اگر حاصل یک عبارت از نوع اعشاری باشد، میتواند به یک متغیر نوع صحیح انتساب داده شود.

۳- اگر حاصل عبارت mark<0 || mark>20 صحیح باشد، یعنی mark خارج از محدوده ۰ تا ۲۰ است

۴- در دستور switch نوع داده عبارتی که داخل پرانتز نوشته می شود، میتواند از هر نوع باشد.

۵- می توانیم حلقه ی while ای بنویسیم که اصلا اجرا نشود.

در سوالات زیر گزینه صحیح را انتخاب کنید. (هر کدام ۰,۲۵ نمره)

۶-در عملگرهای ریاضی کدامیک بالاترین اولویت را دارد؟

الف) + ب) * ج) ٪ د) – (قرینه)

۷-دستور i+=۱ معادل کدامیک از دستورات زیر است؟

الف) i = ۱ ب) i = i + ۱ ج) i ++

۸-فلوچارت زیر نشان دهنده کدام دستور است؟

الف) for ب) while ج) select case د) if

۹- قطعه برنامه زیر چه عملی انجام می دهد؟

```
for (int i = 1 ; i<= 10 ; i=i+2 )

  Console.WriteLine( i);
```

الف) چاپ اعداد ۱ تا ۱۰ ب) چاپ اعداد زوج ۱ تا ۱۰
ج) چاپ اعداد فرد ۱ تا ۱۰ د) چاپ اعداد ۱۰ تا ۱

۱۰- کدام گزینه در مورد حلقه ها درست است؟

الف) سه قسمت درون حلقه for اجباری هستند ب) نوشتن عبارت منطقی حلقه while اجباری است

ج) نوشتن عبارت منطقی حلقه do while اختیاری است د) فقط حلقه while ممکن است بی نهایت بار تکرار شود

در جملات زیر، جاهای خالی را با عبارات مناسب پرکنید.

۱۱- برای محاسبه رقم یکان یک عدد، از عملگر ..... استفاده می شود.

۱۲- برای کاهش تعداد دستورات تکراری برنامه می توان از ..... استفاده کرد.

۱۳- اگر بخواهیم در برخی تکرارهای حلقه دستورات بدنه حلقه انجام نشوند، از دستور ..... استفاده می کنیم.

هر یک از عبارات سمت راست با یکی از عملگرهای منطقی سمت چپ مرتبط است. نام آن را در پرانتز بنویسید.(یک گزینه در ستون پاسخ اضافی است)(۱ نمره)

ا

۱۴- متغیر a بزرگتر مساوی ۱۰ و کوچکتر از ۱۰۰ (---) الف) ||

۱۵- متغیر a بزرگتر از ۱۰۰ یا کوچکتر از ۱۰ (---) ب) !

۱۶- متغیر a بزرگتر از ۱۰۰ نباشد (---) ج) ^

۱۷- متغیر a فقط بزرگتر از ۱۰۰ یا فقط کوچکتر از ۱۰ (---) د) &&

ه) ==

سوالات کوته پاسخ

۱۵- یکتایی یا دوتایی بودن عملگر تفریق را تعیین کنید. (۰,۲۵ نمره)

۱۶- حاصل عبارت زیر چیست؟ (۰,۵ نمره)

```
23.5/ 2
```

۱۷- عبارت ریاضی زیر را با توجه به اولویت عملگرها به صورت یک عبارت قابل قبول در C# بنویسید. (۰,۷۵ نمره)

$$\frac{x+y}{a}$$

۱۸- اگر x دارای مقدار ۳ باشد حاصل عبارت زیر را بدست آورید.(۰,۵نمره)

```
3*++x
```

۱۹- قطعه کد زیر را با switch بازنویسی کنید. (۱ نمره)

```
if (x==1)
        System.Console.WriteLine(1);
else if (x==2)
        System.Console.WriteLine(2);
else if  (x==3)
        System.Console.WriteLine(2);
```

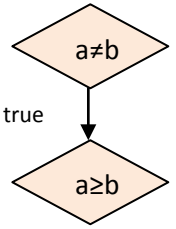
۲۰- چرخهای یک خودرو در حین حرکت خودرو بارها می چرخند. به نظر شما تکرار چرخش چرخ ماشین از کدام نوع حلقه تکرار است؟ (۰,۲۵ نمره)

سوال تشریحی

۲۱- برای تبدیل حلقه for زیر به while چه تعییراتی باید در کد انجام شود؟(۰,۷۵ نمره)

```
for (int a=1; a<=100; a+=5)
    Console.WriteLine(a);
```

۲۲- در شکل زیر معادل هر عبارت مقایسه ای ریاضی را در سی شارپ، جلو آن بنویسید. (۰,۵ نمره)



Trace و خطا یابی

۲۳- عبارت زیر برای محاسبه میانگین سه عدد نوشته شده، اشکال آن را پیدا کرده و تصحیح کنید. (۰,۵ نمره) 15+7+9/3

۲۴- قطعه کد زیر برای تشخیص یک، دو یا سه رقمی بودن یک عدد نوشته شده است. اما با قراردادن عدد ۸ در متغیر number هر سه پیام را نمایش می دهد. با اصلاح کد مشکل را برطرف کنید.(۰,۷۵ نمره)

```
if (number<10)
  Console.WriteLine(" 1 digit");
if (number < 100)
  Console.WriteLine(" 2 digit");
if (number < 1000)
   Console.WriteLine(" 3 digit");
else
Console.WriteLine("out of range");
```

۲۵- در قطعه کد زیر با اینکه مقدار a=۳ است در خروجی عبارتyes و no هر دو نمایش داده میشوند. اشکال آنرا پیدا کرده و تصحیح کنید.(۰,۷۵ نمره)

```
switch(a)
{
    case 3:
        System.Console.WriteLine("yes");
    default :
        System.Console.WriteLine("no");
        break;
}
```

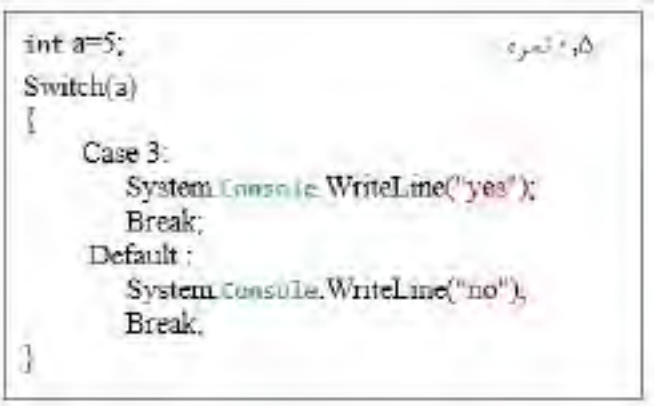
۲۶- حلقه زیر برای نمایش اعداد زوج دورقمی نوشته شده. اما هنگام اجرا در حلقه بی نهایت می افتد. اشکال آنرا برطرف کنید(۰,۵ نمره)

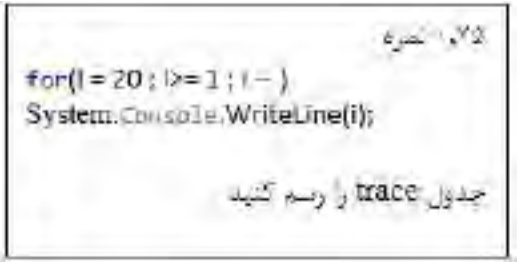
```
for(int i = 10 ; i>= 1 ; i+=2 )
   System.Console.WriteLine(i);
```

۲۷-نتیجه اجرای هر یک از قطعه کدهای زیر چیست؟

(۰,۷۵ نمره)

```
bool a =true, b= false;

System.Console.WriteLine(3>5 ^ a);

System.Console.WriteLine(!b&& 7!=6);
```





int x=4; x*=5; (0.25)

۲۸- نتیجه اجرای هر یک از سه قطعه کد زیر چیست؟(۰/۷۵ نمره)

```
int i=10;
 do
 {
Console.WriteLine(i);
    i *= 2;
  }
 while (i<8);
```

```
for (int a=10;a<8;a*=2)

Console.WriteLine(a);
```

```
int j=10;
while ( j <8)
 {
Console.WriteLine(j);
  j *= 2;
 }
```

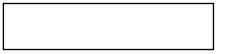
۲۹- خروجی فطعه کد زیر چیست؟(۰,۵ نمره)

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

برنامه نویسی

۳۰- قطعه کد زیر قرار است وضعیت تاهل یک کارمند را بررسی کرده و پیام مناسب با آن چاپ کند. درجای خالی دستور مناسب بنویسید تا قطعه کد به درستی کار کند. (۰,۵ نمره)

```
if (married)
        System.Console.WriteLine(“married”);
```



```
        System.Console.WriteLine(“single”);
```

۳۱- برنامه ای بنویسید که سه عدد از ورودی دریافت کرده و تعیین کند این اعداد می توانند سه ضلع یک مثلث باشند یا نه.(راهنمایی: در مثلث همواره اندازه هر ضلع از مجموع اندازه دو ضلع دیگر کمتر است) (۱,۵ نمره)

۳۲- برنامه ای بنویسید که اعداد فرد یک رقمی را نمایش دهد. از حلقه do .. while استفاده کنید. (۱ نمره)

۳۳- برنامه ای بنویسید که نمره برنامه سازی ۱۵ دانش اموز را دریافت کرده و برای نمرات زیر ۱۲ پیام "تلاش بیشتر" نمایش دهد.(۱,۵ نمره)

ارتباط تمارین فصل و اهداف رفتاری

ردیف	اهداف رفتاری فصل هفتم	خودآزمایی	تمرینات برنامه نویسی
۱.	کاربرد حلقه در برنامه را توضیح دهد.	۱	
۲.	دستورات ایجاد حلقه را نام ببرد و تفاوت هر یک را بیان کند.	۲	
۳.	عملکرد و کاربرد دستور حلقه while را توضیح دهد. ۱و۲و۴و۵	۶	
۴.	عملکرد و کاربرد دستور حلقه for را توضیح دهد.	۱و۲و۳	
۵.	برنامه های کاربردی با حلقه تکرار بنویسد		۲و۳و۴و۵
۶.	در برنامه های خود break و continue را در صورت لزوم به کار بندد.		۱
۷.	ضرورت استفاده از حلقه های متداخل را توضیح دهد و آنها را در برنامه های خود به کار بندد.	۶	۶و۷

بودجه بندی فصل ها

موضوع : دستورات تکرار (حلقه ها)

فصل هفتم
هفته اول اردیبهشت ماه

هدف کلی: آشنایی با کاربرد حلقه ها -عملکرد وکاربرد دستور while

ساعات تئوری: ۳ ساعات عملی: ۵

فعالیتهای آموزشی موازی

-کار عملی با برنامه های نمونه
-استفاده از MSDN (help) برای دیدن ساختار while
- trace کردن حلقه های بدون خروجی توسط دانش آموزان

فعالیتهای رقابتی و ارزشیابی

- ارزشیابی از فصل های۴-۵-۶
- آزمون سرعت و دقت رقابتی:در اختیار قرار دادن ۵ الگوریتم مربوط به حلقه

بودجه بندی فصل ها

فصل هفتم
هفته دوم اردیبهشت ماه

موضوع : دستورات تکرار (حلقه ها)

هدف کلی: آشنایی با عملکرد و کاربرد دستور for

ساعات تئوری: ۳ **ساعات عملی:** ۵

فعالیتهای آموزشی موازی

-بررسی و حل تمرینهای فصل هفتم
- تمرین توسعه از طریق ایده دهی برای حلقه های ساده

فعالیتهای رقابتی و ارزشیابی

- ارزشیابی از عملکرد دستور while(ص-غ و چندگزینه ای و تکمیلی)
- آزمون عملی شبه پایانی (تجسم خلاق)

موضوع : دستورات تکرار (حلقه ها)

هدف کلی: توانایی به کارگیری حلقه های متداخل- دستورات breakوcontinue

ساعات تئوری: ۳ ساعات عملی: ۵

فعالیتهای آموزشی موازی
-بررسی و حل تمرینهای فصل هفتم
-بررسی برنامه با حلقه های پیچیده و متداخل
- تمرکز روی Trace حلقه ها

فعالیتهای رقابتی و ارزشیابی

- ارزشیابی گروهی به منظوررفع اشکال
-تمرین عملی رقابتی از طریق در اختیار قرار دادن کد ناقص انواع حلقه های ساده و متداخل

فصل هفتم
هفته سوم اردیبهشت ماه

آموزش فصل ۷

دستور while صفحات
۱۴۵ و ۱۴۶
دستور do...while
صفحات ۱۵۰و ۱۵۱
مدرس:
خانم جمیله درساره

شهر بندرعباس
استان هرمزگان
زمستان ۱۳۹۴


برد کامپیوتر
FOR
C#
FOR
iF
WHILE
WHILE
سی‌شارپ