طراحی سیستم‌های شیءگرا
با زبان
C#

تالیف

مهندس رمضان عباس‌نژادورزی
مهندس باقر رحیم پورکامی
مهندس ابراهیم هاشمیان

فن‌آوری نوین

سرشناسه	: رمضان عباس‌نژاد ورزی - ۱۳۴۸
عنوان و نام پدیدآور	: طراحی سیستم‌های شی‌گرا با زبان #C/ تالیف رمضان عباس‌نژادورزی، باقر رحیم‌پورکامی، سیدابراهیم هاشمیان.
مشخصات نشر	: بابل: فن‌آوری نوین، ۱۳۹۱.
مشخصات ظاهری	: ۴۳۲ص.: مصور، جدول.
شابک	: ۱۳۰۰۰۰ ریال: 978-600-92254-2-2
موضوع	: سی شارپ (زبان برنامه‌نویسی کامپیوتر)
موضوع	: برنامه‌نویسی شی‌گرا
شماره افزوده	رحیم‌پور کامی، باقر، ۱۳۶۰ -
شماره افزوده	هاشمیان، سیدابراهیم، ۱۳۵۷ -
رده بندی کنگره	: QA۷۶/۷۳/ س۹۵ع۲۵۱۳۹۰
رده بندی دیویی	: ۰۰۵/۱۳۳
شماره کتابشناسی ملی	: ۲۶۶۶۶۳۰

فن‌آوری نوین www.fanavarienovin.net
بابل، صندوق پستی ۴۷۱۶۷-۷۳۴۴۸ تلفن: ۲۲۵۶۶۸۷-۰۱۱۱

طراحی سیستم‌های شیءگرا با زبان #C

تالیف: مهندس رمضان عباس‌نژادورزی- مهندس باقر رحیم‌پور کامی- مهندس ابراهیم هاشمیان

ویراستار: مهندس هادی‌عباسی

ناشر: فن‌آوری نوین

چاپ اول: بهار ۱۳۹۱

جلد: ۲۰۰۰

شابک: ۲ - ۲ - ۹۲۲۵۴ - ۶۰۰ - ۹۷۸

حروفچینی و صفحه‌آرایی: فن‌آوری نوین **قیمت: ۱۳۰۰۰ تومان**

تهران، خ اردیبهشت، نبش وحید نظری، پلاک ۱۴۲ **تلفکس: ۶۶۴۰۰۲۲۰-۶۶۴۰۰۱۴۴**

فهرست مطالب

دیگر آثار مولف			
نام کتاب	انتشارات	نام کتاب	انتشارات
حل مسائل C (مرجع کامل)	فن آوری نوین	آموزش گام به گام Crystal Report	علوم رایانه
حل مسائل C++ (مرجع کامل)	فن آوری نوین	آشنایی با شبکه GSM	علوم رایانه
آموزش گام به گام برنامه نویسی بانک اطلاعات با C# (مرجع کامل)	فن آوری نوین	آموزش گام به گام سیستم عامل لینوکس	علوم رایانه
حل مسائل C# (مرجع کامل)	فن آوری نوین	خود آموز اکسس	علوم رایانه
حل مسائل پاسکال (مرجع کامل)	فن آوری نوین	آموزش گام به گام Word	علوم رایانه
آموزش گام به گام برنامه نویسی بانک اطلاعات با ویژوال بیسیک‌نت (مرجع کامل)	فن آوری نوین	آموزش گام به گام اکسل	علوم رایانه
آموزش گام به گام LINQ با C#	فن آوری نوین	ICDL مهارت ۱: مفاهیم پایه اطلاعات	علوم رایانه
تجارت الکترونیکی	فن آوری نوین	ICDL مهارت۲: به کارگیری کامپیوتر و مدیریت	علوم رایانه
امنیت شبکه	فن آوری نوین	ICDL مهارت۳: واژه پردازی به کمک کامپیوتر	علوم رایانه
اصول طراحی پایگاه‌داده	فن آوری نوین	ICDL مهارت۴: صفحات گسترده	علوم رایانه
آموزش گام به گام دلفی‌نت	علوم رایانه	ICDL مهارت۵: پایگاه داده	علوم رایانه
آموزش گام به گام C#.NET	علوم رایانه	ICDL مهارت۶: ارائه مطلب	علوم رایانه
آموزش گام به گام VisualC++.NET	علوم رایانه	ICDL مهارت۷: اطلاعات و ارتباطات	علوم رایانه
آموزش گام به گام برنامه نویسی با ویژوال C++	علوم رایانه	آموزش گام به گام FLASH MX	علوم رایانه
آموزش گام به گام J#.NET	علوم رایانه	درس و کنکور برنامه‌نویسی به زبان C	علوم رایانه
آموزش گام به گام SQL Server	علوم رایانه	برنامه سازی سیستم	علوم رایانه
آموزش گام به گام SQL	علوم رایانه	پرسش‌های چهار گزینه‌ای پاسکال	علوم رایانه
رهیافت و پرسش‌های چهار گزینه‌ای C	علوم رایانه	آموزش گام به گام VC++	علوم رایانه
رهیافت و پرسش‌های چهار گزینه‌ای C++	علوم رایانه	کارور رایانه ۱	علوم رایانه
تست و پرسش‌های چهار گزینه‌ای C	علوم رایانه	کارور رایانه ۲	علوم رایانه
مبانی فناوری اطلاعات	علوم رایانه	آموزش گام به گام برنامه ویژوال بیسیک‌نت	علوم رایانه
آموزش گام به گام برنامه ویژوال بیسیک	علوم رایانه	برنامه نویسی به زبان اسمبلی	علوم رایانه
برنامه نویسی با دلفی	علوم رایانه		

مقدمه

زبان #C در فناوری دات‌نت (.NET) توسط مایکروسافت ارائه شده است که کاملاً شیءگرا است. امروزه اکثر دانشجویان رشته کامپیوتر با این زبان آشنایی دارند. برنامه‌های متعددی از قبیل تحت کنسول، دسک‌تاپ، بانک‌اطلاعاتی، طراحی صفحات وب، WPF، WCF، تحت شبکه و دستگاه‌های موبایل را می‌توانید با زبان #C بنویسید این کتاب تاکید بیشتری به نوشتن برنامه‌های تحت کنسول نموده است.

از طرف دیگر زبان #C به عنوان سرفصل درس برنامه‌سازی پیشرفته در رشته‌ها کامپیوتر، فناوری اطلاعات، ICT و علوم کامپیوتر تدریس می‌شود.

در حال حاضر کتاب‌های زیادی برای زبان برنامه نویسی #C ارائه شده است که جای تقدیر و تشکر دارد. هر یک از این کتاب‌ها نوع خاص از زبان برنامه‌نویسی #C را مورد بررسی قرار می‌دهند. اما، این کتاب تمرکز بیشتری روی برنامه‌های تحت کنسول دارد.

کتاب حاضر با بیان مسائل متعدد تحت کنسول و حل آن‌ها، دانشجویان محترم را با زبان برنامه‌نویسی #C آشنا می‌کند.

کتاب به صورت گام‌به‌گام به جملات کوتا و ساده بیان گردیده است.

برنامه‌های متن کتاب، مسائل‌حل شده و مسائل حل شده در سایت را می‌توانید از سایت انتشارات فن-آوری نوین به آدرس **www.fanavarienovin.net** دریافت نمایید.

در پایان امیدوارم این اثر مورد توجه جامعه انفورماتیک کشور، اساتید و دانشجویان عزیز قرار گیرد.

مولفین

fanavarienovin@yahoo.com

فصل ۱ آشنایی با زبان C#

پیشرفت‌های زبان‌های برنامه‌نویسی نظیر C++ و جاوا، موجب ایجاد مشکلات و نیازمندی‌های جدیدی گردید. ایجاد یکپارچگی اجزا نرم‌افزاری از زبان‌های مختلف برنامه‌نویسی مشکل بود و در نصب این ابزار مشکلات مشترکی وجود داشت. به همین دلیل بود که نسخه جدید قطعات[1] با نرم‌افزارهای قدیمی سازگار نبود. از طرف دیگر، نیاز به برنامه‌های تحت‌وب، موجب گردید تا NET. و زبان برنامه‌نویسی C# ایجاد شود.

C# زبانی است که برنامه نویسان را قادر می‌سازد، به راحتی بتوانند از زبان‌های مختلف در پروژه‌شان استفاده کنند. زیرا، C# ریشه در C، C++ و جاوا دارد و ابزارهای زیادی از آن‌ها را در خود جمع کرده، علاوه‌بر این، ابزارهای جدیدی به آن‌ها اضافه نموده است و قوانین شی‌گرایی به طور کامل در آن پیاده شده است. در ضمن این زبان با قابلیت برنامه‌نویسی ویژوال، امکان ایجاد برنامه‌هایی با استفاده از **محیط توسعه مجتمع** (IDE)[2] را تامین کرده است. با استفاده از IDE، برنامه‌نویس می‌تواند به راحتی برنامه را ایجاد، اجرا، آزمایش (تست) و رفع اشکال (خطایابی) نماید. پس، در زمان برنامه‌نویسی صرفه‌جویی زیادی خواهد شد. روند ایجاد سریع برنامه‌ها با استفاده از IDE، به **توسعه سریع برنامه** (RAD)[3] معروف است.

مزیت دیگر C# استفاده از قطعات تولیدشده در زبان‌های برنامه‌نویسی مختلف در آن است.

۱ – ۱. فرآیند برنامه‌نویسی در دات‌نت

در طراحی یک برنامه، اولین گام تعیین نوع برنامه‌ای است که می‌خواهید آن را ایجاد کنید. در دات‌نت برنامه‌های متعددی از قبیل برنامه تحت کنسول، برنامه‌های تحت ویندوز، برنامه‌های تحت وب، وب سرویس یا انواع دیگری را می‌توان ایجاد کرد. در این کتاب روش ایجاد برنامه‌های تحت کنسول و تحت ویندوز را می‌آموزیم. گام بعدی انتخاب زبان برنامه‌نویسی می‌باشد. این مرحله از اهمیت ویژه‌ای برخوردار است. چون، زبان‌های غیردات‌نت امکانات مختلفی را در اختیارتان می‌گذارند. اما، در دات‌نت زبان‌های مختلف به یکدیگر شبیه شده‌اند و امکانات یکسانی را در اختیارتان قرار می‌دهند. چون این زبان‌ها در هنگام اجرا از زبان مشترک زمان اجرا (CLR)[4] استفاده می‌کنند. بنابراین، در زمان اجرا این مورد که در طراحی برنامه از چه زبانی استفاده شده است، تفاوتی ندارد. از آنجایی که زبان‌های مختلف گرامرهای متفاوتی دارند، بنابراین باید زبانی انتخاب شود تا با گرامر آن آشنا باشید. چون، گرامر زبان C# شبیه زبان C و C++ است به همین دلیل، زبان برنامه‌نویسی و طراحی را C# انتخاب نمودیم.

---

1 .Components
2. IDE(Integrated Development Environment)
3.RAD(Rapid Application Development)
4. Common Language Runtime

ویژوال استودیودات‌نت از کامپایلرها و زبان‌های مختلفی تشکیل شده‌است که عبارت‌اند از:

۱. ویژوال بیسیک ۲. ویژوال #C

۳. ویژوال ++C ۴. ویژوال #F

علاوه‌بر مایکروسافت شرکت‌های دیگر نیز برای زبان‌های خود کامپایلرهایی را عرضه کرده‌اند که CLR را به عنوان محیط زمان اجرای نهایی مورد استفاده قرار داده‌اند. برخی از این زبان‌ها عبارت‌انداز: APL، Cobol، Fortran، ML، Mercury، Perl، Python، RPG، Smalltalk و غیره.

گام سوم، نوشتن کد مورد نیاز برنامه می‌باشد. البته برنامه‌های مختلف کدهای متعددی خواهند داشت. در ادامه با انواع برنامه و کدهای پیاده‌سازی آن‌ها آشنا خواهید شد. گام‌چهارم، کامپایل نمودن برنامه می‌باشد. کامپایل موجب می‌شود تا خطاهای برنامه (نحوی و گرامری) رفع شده، برنامه به کد میانی CLR ترجمه شود.

۲ – ۱. مجموعه کتابخانه کلاس دات‌نت Framework

علاوه بر CLR، در مجموعه دات‌نت Framework، بخش دیگری به نام کتابخانه کلاس چارچوب (FCL)[1] وجود دارد. این بخش شامل هزاران کلاس می‌باشد که هر کدام وظیفه خاصی دارند. مجموعه‌های FCL و CLR به طراحان اجازه می‌دهند که چندین مدل برنامه را طراحی کنند که عبارت‌اند از:

۱. برنامه‌های تحت کنسول در ویندوز، برنامه‌هایی ایجاد می‌کند که نیاز به رابط گرافیکی کاربر ندارند. این برنامه‌ها از رابط خط فرمان استفاده می‌کنند. این نوع برنامه‌ها معمولاً برای نوشتن ابزارهایی نظیر کامپایلر و بعضی از برنامه‌های کاربردی به کار می‌روند (در فصل‌های ۱ تا ۶ این نوع برنامه‌ها را می‌آموزیم).

۲. برنامه‌های تحت ویندوز، برنامه‌هایی هستند که نیاز به رابط گرافیکی کاربر دارند. برنامه‌هایی تحت ویندوز برنامه‌های دسک‌تاپ نیز نامیده می‌شوند. زمانی که به برنامه‌های تحت وب نیاز نباشد، می‌توان از این برنامه‌ها استفاده نمود (در فصل‌های ۷ تا ۹ روش ایجاد برنامه‌های تحت ویندوز را می‌آموزیم).

۳. برنامه‌های تحت وب، برنامه‌هایی ایجاد می‌کنند که مبتنی بر صفحات HTML[2] هستند. این نوع برنامه‌ها، از طریق سرویس‌دهنده بانک اطلاعاتی یا چندین وب سرویس، اطلاعات مورد نیازشان را دریافت کرده، پردازش‌های مورد نیاز را برروی آن انجام داده، صفحات مبتنی بر HTML ایجاد می‌نمایند تا این صفحات از طریق مرورگرهای وب[3] در کامپیوتر سرویس گیرنده[4] قابل نمایش باشند.

۴. سرویس‌های ویندوز، سرویس‌هایی را می‌توان در دات‌نت ایجاد کرد که توسط مدیر کنترل سرویس ویندوز (SCM)[5] و نیز دات‌نت Framework قابل کنترل هستند. معمولاً این سرویس‌ها برای تبادل اطلاعات بین برنامه‌های مختلف استفاده می‌شوند.

---

[1].Framework Class Library [2]. Hyper Text Markup Language [3]. Browser [4].Client
[5].Windows Service Control Manager

۵. وب سرویس‌ها، سرویس‌ها یا توابعی هستند که به راحتی از طریق شبکه وب قابل دسترسی و فراخوانی هستند.

۶. قطعات و کتابخانه کلاس، در دات‌نت Framework می‌توان قطعات[1] و کتابخانه‌هایی با کلاس‌های جدید ایجاد نمود. این قطعات و کتابخانه‌های کلاس‌های جدید را به راحتی می‌توان در برنامه‌های دیگر (حتی به زبان‌های دیگر) استفاده نمود.

۷. و غیره

۳ – ۱. فضای نام

همان‌طور که بیان گردید، در FCL هزاران کلاس وجود دارند. برای دسته‌بندی کلاس‌ها، تمام کلاس‌های مرتبط به هم در یک فضای نام[2] قرار می‌گیرند. اصلی‌ترین فضای نام، فضای نام System است. این فضای نام، شامل کلاس object و تعدادی کلاس پایه دیگر است. کلاس object یک کلاس پایه است که تمام کلاس‌های FCL از این کلاس مشتق می‌شوند (در فصل ۶ با مفهوم کلاس‌های پایه و مشتق آشنا خواهید شد). کلاس‌هایی که در FCL وجود دارند، **کلاس‌های آماده** نام دارند. علاوه براین کلاس‌ها، برنامه‌نویس می‌تواند کلاس‌های جدیدی را ایجاد کرده و از آن‌ها استفاده کند. چون، ممکن است کلاس موجود در FCL همه نیازهای برنامه‌نویس را برطرف نکند. چگونگی ایجاد این کلاس‌ها را در فصل ۵ و ۶ می‌آموزیم. در این بخش می‌خواهیم به کلاس‌های موجود در FCL بپردازیم. کلاس‌های موجود در FCL با توجه به کاربردشان در فضای‌های نام مختلف قرار می‌گیرند. این عمل دو مزیت زیر را برای برنامه‌نویس در پی دارد:

۱. موجب دسته بندی کلاس‌ها می‌شود. یعنی کلاس‌هایی که به هم مرتبط هستند، در یک فضای نام قرار می‌گیرند تا اولاً بتوان به راحتی آن‌ها را به پروژه اضافه نمود و ثانیاً فضاهای نامی که در پروژه به آن‌ها نیازی نیست، به پروژه اضافه نگردند.

۲. علاوه براین می‌توان در کلاس‌های مختلف از نام‌های تکراری استفاده نمود. فضای نام و کلاس‌ها موجب می‌شوند تا نام‌های تکراری از یکدیگر تفکیک شوند.

وقتی برنامه جدیدی ایجاد می‌کنید فضاهای نام جدول ۱ – ۱ به پروژه اضافه می‌شود (البته وقتی که برنامه‌ای از نوع Console Application ایجاد می‌نمایید. اگر برنامه‌هایی با نوع‌های دیگر به برنامه اضافه کنید، ممکن است فضای‌های نام دیگر به پروژه‌تان اضافه شود). علاوه برفضاهای نامی که به طور خودکار به برنامه اضافه می‌شوند، می‌توانید فضاهای نام دیگر را نیز به پروژه‌تان اضافه کنید. برای این منظور می‌توانید از دستور using به صورت زیر استفاد نمایید:

```
using نام فضای نام;
```

به عنوان مثال، دستورات زیر را ببینید:

```
using System.Convert;
using System.Io;
```

---

[1] .Components [2].Namespace

دستور اول، فضای نام System.Convert را به برنامه اضافه می‌کند تا بتوانید از متدهایی که برای تبدیل انواع داده‌های مختلف به کار می‌روند، استفاده نمایید (این متدها را در ادامه می‌آموزیم) و دستور دوم، فضای نام System.Io را به پروژه اضافه می‌کند تا بتوانید از کلاس‌هایی که جهت ورودی – خروجی داده‌ها مانند فایل‌ها به کار می‌روند، استفاده نمایید.

چنانچه در ابتدای برنامه با دستور using فضای نام را اضافه نکنید در کلیه مکان‌هایی که می‌خواهید از کلاس‌های موجود در آن فضای نام استفاده نمایید باید مسیر کامل فضای نام را ذکر کنید (به صورت زیر):

System.Convert.ToInt32

این دستور از متد ()ToInt32 کلاس Convert موجود در فضای نام System استفاده می‌کند.

هر پروژه جدیدی که ایجاد می‌شود، یک فضای نام جدید همنام با پروژه نیز ایجاد می‌گردد.

۴ – ۱. آموزش زبان‌های برنامه‌نویسی

آموزش زبان‌های برنامه‌نویسی مانند زبان‌های طبیعی زنده دنیا است. یعنی برای آموزش زبان‌های برنامه‌نویسی باید مراحل زیر را انجام داد:

۱. مانند هر زبان طبیعی ابتدا باید علائم تشکیل دهنده زبان را آموخت. به عنوان مثال، زبان فارسی از علائم الف تا ی، ارقام ۰ تا ۹ و علائم خاص مانند ! ، : ، ؟ و غیره تشکیل شده است. هرکدام از این علائم (نمادها) مفهوم خاصی را دارند. زبان #C، نیز از علائم a تا z ، A تا Z ، ۰ تا ۹ ، علائم ویژه نظیر ; ، : ،] ، [، / و غیره تشکیل شده است. ابتدا باید مفاهیم هریک از این علائم را در زبان #C آموخت.

۲. همان‌طور که می‌دانید از ترکیب علائم هر زبان کلمات بوجود می‌آیند. برخی از کلمات دارای معنی و مفهوم هستند و برخی دیگر معنی و مفهوم خاصی ندارند. به عنوان مثال، کلمات **بابا، آب، داد،** در زبان فارسی مفهوم خاصی دارند. ولی کلمات **تپتانم و بکیاپ** مفهوم خاصی ندارند. به کلماتی که در زبان دارای مفهوم خاص هستند، **کلمات کلیدی** می‌گویند. در زبان #C کلمات کلیدی نظیر for، if، else، while و int وجود دارند. در آموزش این زبان ابتدا باید کلمات کلیدی را شناخت. معنی و کاربرد هرکدام از آن‌ها را باید آموخت.

جدول ۱ – ۱ برخی فضاهای نام.

فضای نام	هدف
System	شامل کلاس‌های پایه دات‌نت و انواع داده از قبیل double، int، char و غیره است.
System.Collection.Generic	شامل کلاس‌هایی اصلی کلکسیون در دات‌نت می‌باشد.
System.LINQ	از کلاس‌هایی تشکیل شده است که برای کاربا LINQ [1] به کار می‌روند.
System.Text	از کلاس‌هایی تشکیل شده است که برای کارکردن برروی متن از قبیل رمزگذاری، رمزگشایی، کلاس StringBuilder و غیره به کار می‌روند.

[1]. برای کسب اطلاعات بیشتر در زمینه LINQ به کتاب آموزش گام‌به‌گام LINQ تالیف رمضان عباس‌نژاد ورزی انتشارات فناوری نوین مراجعه فرمایید.

۳. در هر زبان طبیعی از ترکیب کلمات کلیدی با یک قواعد خاص، جمله ایجاد می‌شود (مانند بابا آب داد). همان‌طور که می‌دانید در زبان فارسی ابتدا فاعل، سپس مفعول و در پایان فعل قرار می‌گیرد. در زبان C# نیز برای ایجاد جملات (دستورات) قواعد خاصی وجود دارد. به عنوان مثال، int برای تعریف داده‌های نوع صحیح به کار می‌رود و به صورت زیر استفاده می‌گردد:

int متغیر۱ , متغیر۲ , ... , متغیرn ;

۴. همان‌طور که می‌دانید، در زبان‌های طبیعی از کنار هم قرار گرفتن جملات مرتبط به هم پاراگراف ایجاد می‌شود. در زبان‌های برنامه‌نویسی نیز با کنار هم قرار دادن دستورات مرتبط به هم، بلاک ایجاد می‌شود. در زبان C#، هر بلاک با { شروع و با } خاتمه می‌یابد.

۵. چند پاراگراف صفحات و فصول را ایجاد خواهند کرد و این روند ادامه می‌یابد تا یک کتاب نوشته شود. در زبان‌های برنامه‌سازی نیز نوشتن برنامه‌ها هم همین روند را دارد. تعدادی بلاک، فایل، و چند فایل مرتبط به هم، برنامه را ایجاد می‌کند.

در ادامه کتاب به آموزش زبان C# با این شیوه می‌پردازیم.

۵ – ۱. شناسه‌ها

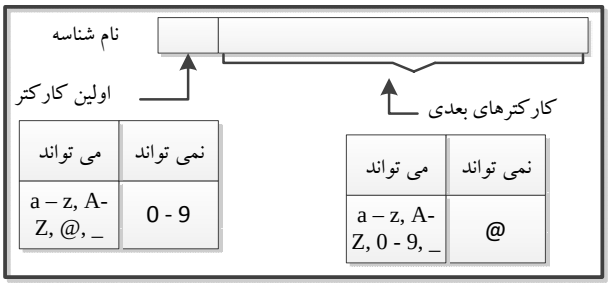
شناسه‌ها[1]، نام‌هایی هستند که برنامه‌نویس به عناصر C# از قبیل کلاس‌ها[2]، فضاهای نام[3]، متدها[4]، فیلدها[5]، خواص[6] و غیره انتخاب می‌کند. به عنوان مثال، (شکل ۱ – ۱) را مشاهده کنید. در این شکل هر کلمه‌ای که در داخل مستطیل قرار دارد، شناسه است. قبل از استفاده از شناسه‌ها باید آن‌ها را نامگذاری نمود. قوانین نامگذاری شناسه‌ها در زیر آمده‌اند (شکل ۱ – ۱). شناسه‌ها در C# با رنگ سبز مشخص می‌گردند.

- کارکترهای الفبایی (A تا Z، a تا z) و خط ربط (_) می‌توانند هر مکان نام شناسه قرار گیرند.

```
using System;

namespace ProjectNamespace {

    class Program {

        static void Main(string[] args) {

            // perform a calculation
            int x = 10 * 10;

            // print out the result of the calculation
            Console.WriteLine("Result: {0}", x);
        }
    }
}
```

شکل ۱ – ۱ برخی از شناسه‌های در یک برنامه.

[1] .Identifiers [2].Classes [3].Namespaces [4].Methods [5]. Fields [6].Properties



شکل ۲ – ۱ روش نامگذاری شناسه‌ها.

- ارقام صفر تا ۹ نمی‌توانند در اولین مکان نام شناسه قرار گیرند، ولی می‌توانند در مکان‌های دیگر نام شناسه مورد استفاده قرار گیرند.
- کارکتر @ می‌تواند در اولین مکان نام شناسه قرار بگیرد، اما، نمی‌تواند در مکان‌های دیگر نام شناسه قرار گیرد.
- نام شناسه نسبت به حروف بزرگ و کوچک حساس است. یعنی، شناسه‌های myVar و MyVar دو نام مختلف برای دو شناسه در نظر گرفته می‌شوند.

۶ – ۱. کلمات کلیدی

کلماتی که در زبان شناخته شده‌اند و مفهوم خاصی در آن زبان دارند، **کلمات کلیدی**[1] نامیده می‌شوند. برخی از کلمات کلیدی را در (شکل ۳ – ۱) می‌بینید. این کلمات در داخل مستطیل قرار دارند. C# از کلمات کلیدی زیادی تشکیل می‌شود که برخی از آن‌ها را در جدول ۲ – ۱ می‌بینید (کلمات کلیدی در برنامه C# با رنگ آبی مشخص می‌شوند).

```
using System;

namespace ProjectNamespace {

    class Program {

        static void Main(string[] args) {

            // perform a calculation
            int x = 10 * 10;

            // print out the result of the calculation
            Console.WriteLine("Result: {0}", x);
        }
    }
}
```

شکل ۳ – ۱ برخی از کلمات کلیدی در برنامه C#.

[1].Keywords

جدول ۲ – ۱ کلمات کلیدی #C.						
abstract	const	extern	out	int	short	typeof
as	continue	false	override	interface	sizeof	uint
base	decimal	finally	params	internal	stackalloc	ulong
bool	default	fixed	private	is	static	uchecked
break	delegate	float	protected	lock	string	unsafe
byte	do	for	public	long	struct	unshort
case	double	foreach	readonly	namespace	switch	using
catch	else	goto	ref	new	this	virtual
char	enum	if	return	null	throw	void
checked	event	implicit	sbyte	object	true	volatile
class	explicit	in	sealed	operator	try	while
کلمات کلیدی مختص زبان #C						
ascending	by	descending	equals	from	get	group
into	join	let	on	orderby	partial	select
set	value	where	yield			

۷ – ۱. فضای سفید

فضای سفید (whitespace)، کارکترهایی هستند که قابلیت چاپ ندارند. این کارکترها توسط کامپایلر نادیده گرفته می‌شوند، اما برنامه‌نویس برای افزایش خوانایی برنامه از این کارکترها در برنامه‌اش استفاده می‌کند. برخی از این کارکترها عبارت‌اند از : ۱. کارکتر فضای خالی (space) ۲. کارکتر Tab ۳. خط جدید (New Line) و ۴. کلید Enter(carriage return).

۸ – ۱. لیترال‌ها

لیترال‌ها[1]، داده‌هایی هستند که به صورت ثابت در کد برنامه‌تان وارد می‌کنید. لیترال‌ها می‌توانند مقادیر عددی، رشته‌ای (که در بین جفت کتیشن قرار می‌گیرند) یا منطقی (True یا False) باشند. در (شکل ۴ – ۱) برخی از لیترال‌ها را می‌بینید. در این شکل لیترال‌ها در داخل مستطیل قرار دارند.

```
using System;

namespace ProjectNamespace {

    class Program {

        static void Main(string[] args) {

            // perform a calculation
            int x = 10 * 10;

            // print out the result of the calculation
            Console.WriteLine("Result: {0}", x);
        }
    }
}
```

شکل ۴ – ۱ برخی از لیترال‌ها در #C.

[1].Literals

۹ – ۱. توضیحات

توضیحات[1] توسط کامپایلر نادیده گرفته می‌شوند و موجب افزایش خوانایی برنامه می‌گردند(شکل ۵ – ۱). در این شکل توضیحات در داخل مستطیل قرار دارند. توضیحات در برنامه با رنگ سبز پررنگ نمایش داده می‌شوند. به دو روش می‌توان توضیحات را به برنامه اضافه کرد:

```
using System;

namespace ProjectNamespace {

    class Program {

        static void Main(string[] args) {

            // perform a calculation
            int x = 10 * 10;

            // print out the result of the calculation
            Console.WriteLine("Result: {0}", x);
        }
    }
}
```

شکل ۵ – ۱ موجب افزایش خوانایی برنامه.

۱. کارکترهای //، تمام کلمات بعد از // **توسط** کامپایلر نادیده گرفته می‌شوند. یعنی، این کلمات توضیحات در نظر گرفته می‌شوند. به عنوان مثال، دستور زیر را در نظر بگیرید:

```
int i = 0;    //Define   I  and initial .
```

این دستور i را تعریف کرده، مقدار صفر را در آن قرار می‌دهد و در ادامه دستور، توضیح برای آن آمده است.

۲. کارکترهای * و / *، توضیحات می‌توانند با کارکترهای * شروع می‌شوند و با کارکترهای / * خاتمه یابند. یعنی، تمام کلماتی که بین * و / * قرار می‌گیرند، توضیحات در نظر گرفته می‌شوند. با این روش می‌توان توضیحات چند سطری(خطی) نیز ایجاد نمود. به عنوان مثال، دستورات زیر را ببینید:

```
/*
This text is ignored by the compiler.
Unlike single-line comments, delimited comments
like this one can span multiple lines.
*/
```

این دستورات توضیحات چند سطری را ایجاد می‌کنند.

در #C نمی‌توانید توضیحات تو در تو تعریف کنید. به عنوان مثال، دستورات زیر را ببینید:

```
/*This is an attempt at a nested comment.
? /*Ignored because it's inside a commentInner comment
? */Closes the comment because it's the first end delimiter
          encountered
? */Syntax error because it has no opening delimiter*/
```

[1].Comments

این دستورات موجب تولید خطا توسط کامپایلر خواهند شد. چون در C# نمی‌توان توضیحات تو در تو ایجاد کرد. اکنون دستورات زیر را ببینید:

```
//Single-line comment      /* Nested comment?
? /*Incorrect because it has no opening delimiter*/
```

این دستورات نیز موجب تولید خطا توسط کامپایلر خواهند شد.

۱۰ – ۱. کارکترهای ویژه

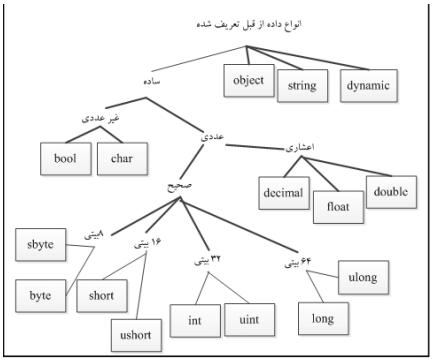
این کارکترها برای نگهداری گروهی از اجزا یا جدا کننده‌ها به کار می‌روند. برخی از این کارکترها را در (شکل ۶ – ۱) می بینید. در این شکل، کارکتر ; انتهای دستورات C# را مشخص می کند. کارکتر }، بلاک C# را باز می کند و کارکتر {، بلاک باز شده C# را می‌بندد. هر بلاک در C# با کارکتر } شروع و با کارکتر { خاتمه می‌یابد. در C# کارکترهای } و { می‌توانند به صورت تو در تو به کار روند(شکل ۵ – ۱ را ببینید).

```
using System;

namespace ProjectNamespace {

    class Program {

        static void Main(string[] args) {

            // perform a calculation
            int x = 10 * 10;

            // print out the result of the calculation
            Console.WriteLine("Result: {0}", x);
        }
    }
}
```

شکل ۶ – ۱ نمایش کارکترهای خاص.

۱۱ – ۱. انواع داده

در C#، دو نوع داده مقدار و ارجاع وجود دارند. انواع داده‌های C# را در (شکل ۷ – ۱) می‌بینید. خلاصه این داده‌ها در جدول ۳ – ۱ آمده‌اند.



شکل۶ – ۱ انواع داده های C#.

جدول ۳ - ۱ انواع داده های C#

نام	هدف	محدوده	معادل .NET	مقدار پیش فرض
sbyte	عدد صحیح ۸ بیتی با علامت	-128….127	System.SByte	**0**
byte	عدد صحیح ۸ بیتی بدون علامت	0….255	System.Byte	**0**
short	عدد صحیح ۱۶ بیتی با علامت	-32768…32767	System.Int16	**0**
ushort	عدد صحیح ۱۶ بیتی بدون‌علامت	0….65535	System.UInt16	**0**
Int	عدد صحیح ۳۲ بیتی با علامت	-2147483648… 2147483647	System.Int32	**0**
uint	عدد صحیح ۳۲بیتی بدون علامت	0….4294964295	System.UInt32	**0**
long	عدد صحیح ۶۴ بیتی با علامت	-922337203685477 5808… 922337203685477 807	System.Int64	**0**
ulong	عدد صحیح ۶۴ بیتی بدون‌علامت	0… 18446744073 709551615	System.UInt64	**0**
float	عدد اعشاری با دقت معمولی	$105 \times 10^{-45} \ldots 3.4 \times 10^{38}$	System.Single	0.0f
double	عدد اعشار ی با دقت مضاعف	$5 \times 10^{-324} \ldots 1.7 \times 10^{308}$	System.Double	0.0d
bool	نوع منطقی	True, False	System.Boolean	False
char	کارکتر یونیکد،	U+0000 …. U+ffff	System.Char	\x0000
decimal	عدد دهدهی با ۲۵ رقم اعشار	$\pm 1.0 \times 10^{28} \ldots \pm 7.9 \times 10^{28}$	System.Decimal	0m
Object	کلاس پایه ای که همه انواع دیگر از آن مشتق می‌شوند	-----	System.object	null
string	مجموعه ای از کارکترهای یونیکد	------	System.string	null

۱۲ – ۱. انواع مقدار

متغیرنوع مقدار (value-type)، به طور مستقیم داده‌اش را نگهداری می‌کند. یعنی، محتوی متغیر نوع مقدار، مقدارش است. به عنوان مثال، دستور زیر مقدار ۲۵ را به متغیری به نام i نسبت می‌دهد:

```
int   i = 25;
```

این عمل به شکل مقابل انجام می‌شود:

نوع	نام	مقدار
int	i	25

وقتی متغیر نوع مقدار ذخیره می‌شود، #C نوشته‌ای از نوع، نام شناسه و مقدار را نگهداری می‌کند و هنگامی که متغیر نوع مقدار را کپی می‌کنید، متغیر دوم جداگانه‌ای با همان نوع و مقدار ایجاد می‌شود. به عنوان مثال، دستور زیر کپی i را در j نشان می‌دهد:

```
int   j = i;
```

این عمل به شکل مقابل انجام می‌گردد:

نوع	نام	مقدار
int	i	25
int	j	25

این متغیرها ارتباطی با هم ندارند. اگر مقدار متغیری عوض شود، متغیر دیگر مقدار قبلی‌اش را نگهداری می‌کند. به عنوان مثال، دستور مقابل را ببینید:

```
i = 50;
```

این دستور، مقدار i را به ۵۰ تغییر می‌دهد، اما، مقدار j همان ۲۵ باقی خواهد ماند (شکل مقابل را مشاهده کنید):

نوع	نام	مقدار
int	i	50
int	j	25

تمام انواع تعریف شده در جدول۴ – ۱ (به جز نوع object و string) متغیرهای از نوع مقدار را تعریف می‌کنند.

۱۳ – ۱. انواع ارجاع

انواع ارجاع (Reference Types) به دو شکل یک شی و ارجاع به آن شیء می‌باشند. به عنوان مثال، دستورات زیر را ببینید:

```
StringBuilder  obj1 = new StringBuilder ("نوع ارجاع");
```

نوع StringBuilder برای تعریف رشته‌ای از کارکترها به کار می‌رود. این دستور به شکل زیر عمل می‌کند:

نوع	نام	ارجاع به
StringBuilder	Obj1	

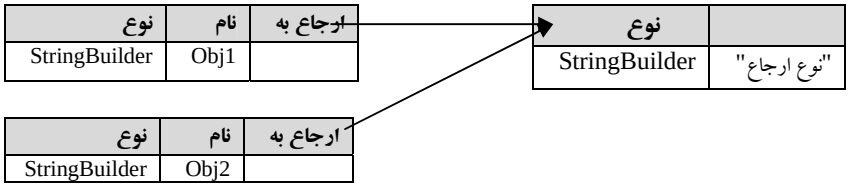
نوع	مقدار
StringBuilder	نوع ارجاع

همان‌طور که در این شکل می‌بینید، نمی‌توان به طور مستقیم به شیء و مقدار آن دستیابی داشت. برای انجام این کار باید از طریق ارجاع استفاده نمایید. در این شکل متغیر ارجاع و شیء را مشاهده می‌کنید. ارجاع، شامل نوع متغیر، نام آن و پیوند که آن را به شیء ارجاع می‌دهد و شیء شامل مقدار است.

وقتی که متغیری از نوع ارجاع را کپی می‌کنید، کپی جدیدی از ارجاع ایجاد خواهد شد، اما، یک شیء جدید نیست. به عنوان مثال، دستور زیر را مشاهده کنید:

```
StringBuilder obj2 = obj1;
```

این دستور، بیان می‌کند که obj2 و obj1 هر دو شی به مقدار "نوع ارجاع" اشاره می‌کنند(شکل زیر را ببینید):



همان طور که در این شکل می‌بینید، فقط یک شیء از نوع StringBuilder داریم، اما دو ارجاع به نام‌های obj2 و obj1 داریم که به آن اشاره می‌کنند و بر خلاف نوع مقدار، مقدار هر دو ارجاع یکی است(یعنی، مقدار "نوع ارجاع" را دارند). با تغییر مقدار یکی، مقدار دیگری نیز تغییر خواهد کرد. C# دو نوع ارجاع اولیه به نام‌های String و object دارد.

دو ارجاع به یک شیء می توانند انواع مختلف داشته باشند.

متغیرهای نوع ارجاع را می‌توان طوری تعریف کرد که به جایی اشاره نکنند. به عنوان مثال، دستور زیر را در نظر بگیرید:

```
StringBuilder   obj = null;
```

نوع	نام	ارجاع به
StringBuilder	obj	null

انواع تهی پذیر

انواع تهی پذیر[1]، اجازه می‌دهند تا متغیری از نوع مقدار ایجاد کنید که می‌تواند یک داده معتبر و غیر معتبر را بپذیرد. بنابراین می‌توانید قبل از استفاده از متغیر به معتبر بودن داده آن مطمئن شوید. یک نوع تهی پذیر همیشه بر پایه نوع دیگر است. ایجاد نوع تهی پذیر به صورت زیر است:

نوع پایه ؟ نام متغیر ;

به عنوان مثال، دستور زیر را در نظر بگیرید:

```
int?    i = 20;
```

این دستور متغیر i را از نوع تهی‌پذیر تعریف کرده مقدار ۲۰ را در آن قرار می‌دهد.

اکنون با عملگر =! می‌توان مقدار i را با تهی مقایسه کرد. دستور زیر را در نظر بگیرید:

```
bool      b = i != null;
```

---

[1].Nullable Types

این دستور b را از نوع bool تعریف می‌کند. اگر i برابر تهی باشد، True در b، وگرنه False در b قرار می‌گیرد.

به سه طریق می‌توان به متغیر از نوع تهی پذیر مقدار داد که عبارت اند از:

۱. مقداردهی در هنگام تعریف

۲. مقداردهی از طریق یک متغیر تهی‌پذیر دیگر

۳. مقدار null

به عنوان مثال، دستور زیر را مشاهده کنید:

```
int? myI1, myI2, myI3;
myI1 = 28;          // Value of underlying type
myI2 = myI1;       // Value of nullable type
myI3 = null;      // Null
```

دستور اول، سه متغیر به نام‌های myI1 , myI2 , myI3 از نوع تهی پذیر تعریف می کند، دستور دوم، مقدار ۲۸ را در myI1 قرار می‌دهد، دستور سوم، مقدار متغیر تهی پذیر myI1 را در متغیر تهی پذیر myI2 قرار می‌دهد و دستور چهارم، مقدار (null) را در متغیر myI3 قرار می‌دهد.

دستورات

دستورات در زبان C# دو نوع‌اند:

۱. **دستورات ساده**، هر دستور که با یک ; خاتمه می‌یابد، **دستور ساده** (Simple statement) نام دارد. به عنوان مثال، دستور زیر را مشاهده کنید:

```
int   i = 5 ;
```

این دستور، متغیر i را از نوع int تعریف کرده، مقدار اولیه ۵ را در آن قرار می‌دهد.

۲. **بلاک**، مجموعه‌ای از صفر یا چند دستور مرتبط به هم که در داخل بلاک باز ({) و بلاک بسته (}) قرار می‌گیرند، را **بلاک** گویند.

به عنوان مثال، دستورات مقابل را ببینید:

```
{
  int  i = 5;
  int  j = 10;
   …
}
```

این دستورات تشکیل بلاک را می‌دهند.

متغیر

برای نگهداری هرچیزی لازم است که از یک ظرف متناسب با آن استفاده نمود. به عنوان مثال، در خانه برای نگهداری مواد غذایی، ظروف مختلفی وجود دارند که هرکدام برای نگهداری مواد خاصی به کار می‌روند. مثلاً، بطری برای نگهداری آب و... به همین ترتیب در برنامه‌نویسی، برای نگهداری مقادیر از ظروف مخصوصی استفاده می‌شود. ظرف نگهداری داده در زبان‌های برنامه‌نویسی **متغیر** نام دارد. **بنابراین، متغیر نامی است برای یک مکان از حافظه که ممکن است که در طول اجرای برنامه مقدار آن تغییر کند. ولی، در یک لحظه فقط یک مقدار را دارد.**

برای استفاده از متغیرها سه عمل باید انجام شود که عبارت‌اند از:

۱. نام‌گذاری متغیرها

بعد از این که یک بچه بدنیا آمد، برای شناسایی او نامی انتخاب می‌کنید. جهت مراجعه به متغیرها نیز از نام آن‌ها استفاده می‌شود. برای نام‌گذاری بچه‌ها ثبت احوال از قوانینی خاصی پیروی می‌کند، به عنوان مثال، اجازه نمی‌دهد نام بچه را **رضا۱** انتخاب کرد. در زبان C# نیز برای نام‌گذاری متغیرها قوانین زیر وجود دارد:

۱. نام متغیر می‌تواند ترکیبی از حروف a تا z یا A تا Z ارقام، خط ربط (–)، ارقام ۰ تا ۹ باشد.

۲. حرف اول متغیر نمی‌تواند ارقام ۰ تا ۹ باشد.

۳. نام متغیر می‌تواند دارای هر طولی باشد. ولی، C#، ۳۱ حرف اول نام را برای متغیر در نظر می‌گیرد.

۴. زبان C# بین حروف بزرگ و کوچک فرق می‌گذارد. یعنی، متغیرهای Count و count با هم فرق دارند.

۵. نام متغیر نمی‌تواند از کلمات کلیدی یا نام توابع انتخاب شود. برخی از نام‌های مجاز برای متغیر عبارت‌اند از: area، sum، sum1، pr_1 و .... اما، نام‌های زیر برای متغیر مجاز نیستند:

- **نام 2test:** نام متغیر نمی‌تواند با ارقام 0 تا ۹ شروع شود.
- **نام $test:** در نام متغیر نمی‌توان از $ استفاده کرد.
- **نام store 2:** در نام متغیر نمی‌توان از کارکتر فاصله[1] استفاده کرد.
- **نام .jpg:** نام متغیر نمی‌تواند با کارکتر نقطه (.) شروع شود.

۲. معرفی متغیرها

همان‌طور که بیان گردید، هر ظرفی برای نگهداری نوعی غذا به کار می‌رود. بنابراین، متغیرها نیز باید دارای نوع باشند تا بتوانند انواع داده‌ها را ذخیره کنند. چون داده‌ها دارای انواع مختلف هستند، بنابراین متغیرها که داده‌ها را نگهداری می‌کنند، باید دارای نوع باشند. نوع متغیر تعیین می‌کند اولاً چه نوع داده‌ای می‌تواند در آن متغیر قرار گیرد و ثانیاً، این متغیر به چند بایت از حافظه نیاز دارد. تعیین نوع متغیر به صورت زیر می‌باشد:

نوع داده‌ای لیست متغیرها ;

نوع داده‌ای، یکی از انواع داده بیان شده نظیر int، float، double و غیره در C# می‌باشد. اگر تعداد متغیرها بیش از یکی باشند، با کاما (,) از هم جدا می‌شوند.

مثال ۱–۱. دستورات زیر متغیرهای a، b و c را از نوع int، d را از نوع double، f_1 و f_2 را از نوع float تعریف می‌کنند.

```
int a, b, c;
double1 d;
float  f1, f2;
```

در این تعریف متغیرهای a، b و c هرکدام چهار بایت، d، ۸ بایت و متغیرهای f_1 و f_2 هریک ۴ بایت از حافظه را اشغال می‌کنند.

[1] .blank(space)

۳. مقداردهی به متغیرها

هر متغیر دارای **نام، نوع، اندازه** و یک **مقدار** است.

سه روش برای مقداردهی به متغیرها وجود دارد، یکی از روش‌ها، مقداردهی به متغیرها در هنگام تعریف آن است.

مثال ۲ – ۱. دستورات زیر متغیرهای a و b را از نوع int تعریف کرده، مقادیر ۱۰ و ۱۲ را به آن‌ها تخصیص می‌دهند و متغیر PI را از نوع float با مقدار 3.14 تعریف می‌کنند.

```
int x = 10 , y = 12;
float  PI = 3.14;
```

بعد از تعریف متغیر نیز می‌توان به آن‌ها مقدار داد. برای این منظور می‌توانید از دستور انتساب (عملگر =) یا دستورات ورودی استفاده کنید. نمونه‌ای از کاربرد عملگر = در زیر آمده است. در ادامه، دستورات ورودی جهت تخصیص مقدار به متغیرها را می‌بینید.

مثال ۳ – ۱. دستوراتی که سه متغیر به نام‌های a، f و b1 را به ترتیب از نوع int، float و bool تعریف می‌کنند و سپس، به آن‌ها مقادیر ۱۰، ۱۳,۷ و false را تخصیص می‌دهند.

```
int a;
float f;
bool b1;
a = 10;
f = 13.7;
b1 = false;
```

نکته: وقتی مقدار جدیدی در متغیر قرار می‌گیرد، این مقدار جایگزین مقدار قبلی می‌شود. یعنی، مقدار قبلی از دست می‌رود (حذف می‌گردد).

۱۴– ۱. ثابت‌ها

ثابت شناسه‌ای (نام خانه‌ای از حافظه) است که مقدار آن در طول اجرای برنامه تغییر نمی‌کند. ثابت‌ها انواع مختلف دارند. ثابت‌ها می‌توانند عددی صحیح، اعشاری، کارکتری، رشته‌ای یا منطقی باشند. ثابت‌های کارکتری بین تک کتیشن (') قرار می‌گیرند (مانند 'C')، ثابت‌های رشته‌ای بین جفت کتیشن قرار می‌گیرند (نظیر "C#") و ثابت‌های منطقی مقادیر true یا false هستند. به عنوان مثال، مقدار 3.1415، (عدد∏) را مشخص می‌کند. این عدد یا هر ثابت دیگر ممکن است چندین مرتبه در برنامه استفاده شود. به دلیل راحتی اصلاح و تغییر مقدار ثابت‌ها به آن‌ها نام تخصیص می‌دهند. تعریف ثابت در C# به صورت‌های زیر انجام می‌شود:

1.const نوع ثابت نام ثابت = مقدار ثابت ;

به عنوان مثال، دستور مقابل را ببینید:

```
const float  PI = 3.1415;
```

این دستور ثابت PI را با مقدار 3.1415 تعریف می‌کند. اکنون دستور مقابل را ببینید.

```
PI = 3.141505;
```

این دستور نادرست است. زیرا، مقدار ثابت‌را نمی‌توان تغییر داد.

۱۵ – ۱. عملگرها

عملگرها[1]،نمادهایی هستند که اعمال خاصی را برروی داده انجام می‌دهند. عملگرها انواع مختلف دارند که برخی از آن‌ها عبارت‌اند از:

---

[1] .operators

۱. عملگرهای محاسباتی **۲.** عملگرهای رابطه‌ای (مقایسه‌ای)
۳. عملگرهای ترکیبی **۴.** عملگرهای منطقی
۵. عملگرهای خاص

جدول ۴ – ۱ عملگرهای محاسباتی.

عملگر	نام عملگر	مثال	نتیجه	توضیحات
+	جمع	۳ + ۱۲	۱۵	عملوند اول را با عملوند دوم جمع می‌کند.
-	تفریق	۳ – ۱۳٫۵	۱۰٫۵	عملوند دوم را از عملوند اول کم می‌کند.
*	ضرب	۱۲ * ۲٫۵	۳۰	عملوند اول را در عملوند دوم ضرب می‌کند.
/	تقسیم	۱۳ / ۲	۶	عملوند اول را بر عملوند دوم تقسیم می‌کند.
%	باقی‌مانده تقسیم صحیح	۱۳ ٪ ۵	۳	باقی‌مانده تقسیم صحیح عملوند اول بر عملوند دوم را محاسبه می‌کند.
++	افزایش	x = 10; x++;	۱۱	یک واحد به عملوند اضافه می‌کند.
--	کاهش	x = 10; x --;	۹	یک واحد از عملوند کم می‌نماید.

۱ – ۱۵ – ۱. عملگرهای محاسباتی

این عملگرها برای انجام محاسبات برروی داده‌های عددی به کار می‌روند (جدول ۴ – ۱). از جمله این عملگرها می‌توان عملگرهای + (جمع)، - (تفریق)، * (ضرب)، / (تقسیم)، ٪ (باقی مانده تقسیم صحیح)، ++ (افزایش) و -- (کاهش) را نام برد. عملکرد عملگرهای +، -، * و / را از قبل می‌دانید. عملگر ٪ برای محاسبه باقی‌مانده تقسیم صحیح به کار می‌رود.

مثال ۴ – ۱. دستورات زیر ابتدا ۱۰ را در x، ۳ را در y قرار می‌دهند و در پایان، باقی‌مانده تقسیم صحیح ۱۰ بر ۳ را (یک) در z قرار می‌دهند.

```
int x = 10, y = 3;
int z = x % y;
```

در دستور دوم، مقدار متغیرهای x و y دست نخورده باقی می‌مانند.

عملگر ++ یک واحد به محتویات عملوند اضافه می‌کند. اما، عملگر -- یک واحد از محتویات عملوند کم خواهد کرد. چنانچه عملگرهای ++ و -- قبل از عملوند قرار گیرند، ابتدا یک واحد به محتویات عملوند اضافه کرده یا از آن کسر می‌کند. سپس، عبارت ارزیابی می‌گردد (مثال زیر را ببینید).

مثال ۵ – ۱. پس از اجرای دستورات زیر مقدار متغیرهای x و y چیست؟

```
int  x = 8;
y = --x;
```

دستور اول، x را تعریف کرده، مقدار ۸ را در آن قرار می‌دهد. دستور دوم، ابتدا یک واحد از x کم کرده (x = x – 1)، بنابراین x برابر با ۷ خواهد شد. سپس، مقدار x را در y قرار می‌دهد. یعنی، ۷ را در y قرار خواهد داد.

اگر عملگرهای + + یا - - بعد از عملوند قرار گیرند، ابتدا عبارت ارزیابی خواهد شد و سپس یک واحد به مقدار عملوند اضافه خواهد شد یا از آن کم می‌گردد (به مثال زیر دقت کنید).

مثال ۶ – ۱. پس از اجرای دستورات زیر مقادیر متغیرهای x و y چیست؟

```
int  x =10;
y = x++;
```

دستور اول، ابتدا x را تعریف کرده، مقدار ۱۰ را در آن قرار می‌دهد. دستور دوم ابتدا، مقدار x را در y قرار می‌دهد (y = x)، بنابراین y برابر ۱۰ خواهد شد و سپس، به x یک واحد اضافه می‌کند. یعنی، ۱۱ در x قرار می‌گیرد.

۲ – ۱۵ – ۱. عملگرهای رابطه‌ای (مقایسه‌ای)

این عملگرها برای مقایسه دو عملوند به کار می‌روند و نتیجه درست یا نادرست را برمی‌گردانند. عملگرهای رابطه‌ای (مقایسه‌ای) در جدول ۵ – ۱ آمده‌اند. دقت کنید عملگر = = تساوی (مساوی بودن) می‌باشد.

جدول ۵ – ۱ عملگرهای رابطه‌ای (مقایسه‌ای).

عملگر	نام عملگر	مثال	نتیجه	توضیحات
>	بزرگتر	۲ > ۳	false	اگر عملوند اول بزرگتر از عملوند دوم باشد، نتیجه درست است، وگرنه نتیجه نادرست می‌باشد.
>=	بزرگتر یا مساوی	۳ =< ۵	true	اگر عملوند اول بزرگتر یا مساوی عملوند دوم باشد، نتیجه درست است، وگرنه، نتیجه نادرست می‌باشد.
<	کوچکتر	۵ < ۷	true	اگر عملوند اول کوچکتر از عملوند دوم باشد، نتیجه درست است، وگرنه نتیجه نادرست است.
<=	کوچکتر یا مساوی	۵ =< ۳	false	اگر عملوند اول کوچکتر یا مساوی عملوند دوم باشد، نتیجه درست است، وگرنه نتیجه نادرست خواهد شد.
=!	نامساوی	۲ =! ۵	true	اگر عملوند اول مخالف عملوند دوم باشد، نتیجه درست است، وگرنه، نتیجه نادرست خواهد بود.
==	تساوی	۲ == ۳	false	اگر عملوند اول مساوی عملوند دوم باشد، نتیجه درست است، وگرنه نتیجه نادرست خواهد شد.

۳ – ۱۵– ۱. عملگرهای ترکیبی

این عملگرها، ترکیبی از عملگرهای محاسباتی و = هستند. عملکرد این عملگرها را در جدول ۶ – ۱ می‌بینید.

۴ – ۱۵ – ۱. عملگرهای منطقی

عملگرهای منطقی، برروی عبارات منطقی درست یا نادرست عمل می‌کنند. نتیجه عملگرهای منطقی در جدول ۷– ۱آمده است. همان‌طور که در جدول ۷ – ۱ می‌بینید، هنگامی نتیجه عملگر && (**و** منطقی) درست

است که هر دو عملوند نتیجه درست داشته باشند. اما نتیجه عملگر || (**یا** منطقی) هنگامی نادرست است که هر دو عملوند نادرست باشند.

جدول ۶- ۱ عملگرهای ترکیبی.

عملگر	روش استفاده	مثال	نتیجه	عملکرد
+=	x += y;	x = 3; x += 5;	8	x = x + 5;
-=	x -= y;	x = 7; x -= 3;	4	x = x - 3;
*=	x *= y;	x = 3; x *= 5;	15	x = x * 5;
/=	x /= y;	x = 17; x /= 5;	3	x = x / 5;
%=	x %= y;	x = 17; x %= 5;	2	x = x % 5;

جدول ۷ – ۱ عملکرد عملگرهای منطقی.

x	y	x && y	x || y	!x	!y
false	false	false	false	true	true
true	false	false	true	true	false
false	true	false	true	false	true
true	true	true	true	false	false

مثال ۷ – ۱. نتیجه اجرای دستورات زیر چیست؟

```
int x = 10, y = 15;
bool  b, b1;
b = (x == 15) || (y > 7);
b1 = (x > 5) && (y == 15);
```

دستور اول، x و y را از نوع int تعریف کرده، مقادیر ۱۰ و ۱۵ را به ترتیب به x و y تخصیص می‌دهد. دستور دوم، متغیرهای b و b1 را از نوع منطقی (bool) تعریف می‌کند. دستور سوم، مقدار true را در b قرار می‌دهد. زیرا، نتیجه x == 15، false می‌باشد. اما، نتیجه y > 7، true است. بنابراین، نتیجه false || true برابر با true می‌باشد. دستور چهارم، مقدار true را در b1 قرار خواهد داد. زیرا، نتیجه عبارت (x > 5) برابر با true است. اما، نتیجه عبارت y == 15، true می‌باشد. در نتیجه حاصل عبارت true && true برابر true است.

۵ – ۱۵ – ۱. عملگرهای خاص

علاوه بر عملگرهای بیان شده، برخی از عملگرها در #C کاربرد خاصی دارند. این عملگرها را در زیر می‌بینید:

۱. **عملگر؟:** برای بررسی شرط خاصی به کار می‌رود و به صورت زیر استفاده می‌شود:

; عبارت ۲ : عبارت ۱ ? (شرط)

در این ساختار ابتدا شرط ارزیابی می‌شود (شرط می‌تواند یک شرط ساده یا ترکیبی باشد)، اگر نتیجه ارزیابی شرط درست باشد، **عبارت۱** انجام می‌شود، وگرنه، **عبارت۲** انجام خواهد شد (مثال زیر را ببینید).

مثال ۸ – ۱. نتیجه اجرای دستورات زیر چیست؟

```
int x = 10, y = 15;
int z = (x > y || y < 17) ? ++x : ++y;
```

دستور اول، x و y را از نوع int تعریف کرده، مقادیر ۱۰ و ۱۵ را به ترتیب به x و y تخصیص می‌دهد. دستور دوم، ابتدا نتیجه عبارت شرطی (x > y || y <17) را بررسی می‌کند که درست می‌باشد. زیرا، ۱۵ < ۱۰ نیست، اما ۱۷ > ۱۵ است. بنابراین، true || false برابر true می‌باشد. پس، عبارت ۱ (یعنی، x++) انجام خواهد شد. یعنی، ابتدا به x یک واحد اضافه می‌شود (مقدار x برابر با ۱۱ خواهد شد) و مقدار ۱۱ در y قرار می‌گیرد.

۲. عملگر کاما (,): برای انجام چند عبارت به صورت پی در پی از عملگر کاما به صورت زیر استفاده می‌شود:

(عبارت n , ... , عبارت ۲ , عبارت ۱)

در این ساختار، عبارت‌های ۱ تا n به ترتیب انجام خواهند شد.

مثال ۹ – ۱. نتیجه اجرای دستورات زیر چیست؟

```
int x = 5, y = 3;
(x ++, y += x, x += y)
```

دستور اول، متغیرهای x و y را از نوع int تعریف می‌کند. در دستور دوم به ترتیب عبارات x ++، y += x و x += y انجام خواهند شد. یعنی، ابتدا، به مقدار x یک واحد اضافه می‌شود (x برابر ۶ خواهد شد). سپس، عبارت y += x انجام می‌گردد. یعنی، y برابر با ۳ + ۶ (۹) خواهد شد و در پایان، عبارت x += y انجام می‌گردد. یعنی، x برابر با ۹ + ۶ (۱۵) خواهد گردید.

۳. عملگر sizeof: تعداد بایت‌هایی که یک نوع یا یک متغیر اشغال می‌کند را مشخص می‌کند و به صورت‌های زیر به کار می‌رود:

sizeof (نوع داده);
sizeof نام متغیر;

مثال ۱۰ – ۱. نتیجه اجرای دستورات زیر چیست؟

```
int a, b;
a = sizeof (double);
b = sizeof a;
```

دستور اول، متغیرهای a و b را از نوع int تعریف کرده، دستور دوم تعداد بایت‌هایی که یک نوع double اشغال می‌کند (یعنی، ۸) را a قرار می‌دهد و دستور سوم، مقدار فضایی که متغیر a اشغال می‌نماید (یعنی، ۴) را در b قرار می‌دهد. چون، متغیر a از نوع int می‌باشد.

۴. عملگر = (انتساب): برای انتساب مقادیر به متغیرها یا مقدار یک متغیر به متغیر دیگر به کار می‌رود و به صورت زیر استفاده می‌شود:

نام متغیر = مقدار ;

مثال ۱۱ – ۲. نتیجه اجرای دستورات زیر چیست؟

```
int x = 5;
int y = x ++ * 5;
```

دستور اول، x را از نوع int تعریف کرده، ۵ را در آن قرار می‌دهد. دستور دوم، ابتدا مقدار x (۵) را در ۵ ضرب کرده، در y قرار می‌دهد (یعنی، ۲۵ را در y قرار می‌دهد) و سپس، به x یک واحد اضافه می‌نماید (یعنی، x برابر ۶ خواهد شد).

۵. عملگر (): برای تغییر اولویت‌های عملگرها در عبارت به کار می‌رود.

انتساب می‌تواند چندگانه باشد. در این صورت انتساب به صورت زیر انجام خواهد شد:

متغیر ۱ = متغیر ۲ = ... = متغیر n = مقدار;

به عنوان مثال، دستورات زیر a، b و n را از نوع int تعریف کرده، مقدار ۱۰ را به آن‌ها تخصیص می‌دهند:

```
int a, b, n;
a = b = n = 10;
```

جدول ۸ – ۱ تقدم عملگرها در C#

نوع عمل	عملگر(ها)	شرکت پذیری
عملیات ابتدایی	(x), x.y, f(x), a[x], x++, x--, new, typeof, sizeof, checked, unchecked	از چپ
عملیات یکانی	+, -, !, ~, ++x, --x, (T)X	از چپ
عملیات ضربی	*, */, %	از چپ
عملیات جمعی	+, -	از چپ
عملیات شیفت	<< , >>	از چپ
عملیات رابطه‌ای	<, >, <=, >=, is	از چپ
عملیات تساوی	==, ! =	از راست
عمل و منطقی	&	از چپ
عمل یا منطقی	|	از چپ
عمل xor منطقی	^	از چپ
عمل و شرطی	&&	از چپ
عمل یا شرطی	||	از چپ
عمل شرطی	? :	از چپ
عمل انتساب	= , *=, /=, %=, -=, <<=, >>=, &=, ^=, |=	از راست
جداکننده	,	از چپ

۱۶ – ۱. اولویت عملگرها

فرض کنید عبارت مقابل را داشته باشید: **۲ + ۳ * ۵**

نتیجه این عبارت می‌تواند یکی از مقادیر ۲۵ یا ۱۷ باشد. چون اگر عملگر جمع اول انجام شود، نتیجه عبارت برابر ۳ + ۲ یعنی ۵ و سپس ۵ * ۵ (برابر با ۲۵) خواهد بود. اما، اگر عملگر ضرب اول انجام شود. یعنی ۵ * ۳ ابتدا انجام شود، نتیجه برابر با ۱۵ + ۲ (۱۷) خواهد بود. چون، اولویت عملگر ضرب بیشتر از عملگر جمع می‌باشد سپس نتیجه ۱۷ خواهد شد. اگر عبارت از ترکیبی از عملگرهای مختلف تشکیل شده باشد اولویت عملگرها را طبق جدول ۸– ۱ می‌توان تعیین کرد. یعنی، بالاترین اولویت را عملگر () و پایین‌ترین اولویت را عملگر کاما دارد.

مثال ۱۲ – ۱. تقدم انجام عملگرها در عبارت زیر را تعیین کنید؟