

فهرست

۵۵	if_Elseif ساختار	۲۳	سرآغاز
۵۶	Case دستور	فصل اول : بررسی مقدمات زبان برنامه نویسی	
۵۷	حلقه های تکرار	۲۷	VB.NET
۵۸	For حلقه	۲۹	VB.NET2010 مقدمات زبان برنامه نویسی
۶۱	While حلقه	۲۹	کلمات کلیدی
۶۲	آرایه چیست ؟	۲۹	شناسه
۶۳	آرایه بدون پهنای باند	۳۰	متغیر چیست ؟
۶۴	For Each حلقه	۳۰	نامگذاری متغیر طبق قوانین
۶۶	شکستن دستورات	۳۱	تعیین نوع متغیر
۶۷	ادامه خطوط به صورت ضمنی	۳۲	انواع داده ها
۶۸	توضیحات	۳۲	داده های عددی
۶۹	فصل دوم : زیر برنامه ها و محدوده متغیرها	۳۳	داده های غیر عددی
۷۱	Sub روال	۳۴	چگونگی تعریف متغیر
۷۴	پارامترها	۳۶	انتساب داده به متغیر
۷۵	ByRef , By Val	۳۸	استفاده از نوع داده های کاراکتری و رشته ای
۷۶	توابع (Function)	۳۸	داده های کاراکتری
۸۰	تکنیک مقداردهی به پارامترها بدون تقدم	۳۸	داده های رشته ای
۸۰	پارامترهای اختیاری (Optional)	۳۹	ثابتها
۸۲	عبارات لاندا (Lambda Expressions)	۴۰	کلمه کلیدی Nothing
۸۳	متغیرهای محلی و سراسری	۴۰	نوع Nullable
	حق دسترسی متغیرهای سراسری برای زیربرنامه ها	۴۱	داده های شمارشی
۸۳		۴۳	نوع داده های ساختاری
	متغیرهای محلی در ساختارهای گردش و شرطی	۴۴	کلمه کلیدی With
۸۶		۴۴	آشنایی با محیط Console Application
۸۸	متغیر محلی Static	۴۶	شروع کدنویسی در Console
۸۸	مستند کردن توابع توسط XML	۴۶	دستورات ورودی /خروجی
	تولید خودکار XML Documentation Comments	۴۸	عملگرها
۸۹		۵۳	اولویت عملگرها
۹۱	تقسیم بندی کدها	۵۳	دستورات شرطی
۹۱	ایجاد محدوده به طور دلخواه (Region)	۵۳	ساختار دستور IF
۹۳	فصل سوم : مقدمات Linq	۵۵	ساختار if_Else نردبانی



۱۲۲	کد کردن کلاسها
۱۲۵	اضافه کردن متدها
۱۲۶	متد Constructor
۱۲۸	نمونه گیری (Instance)
۱۳۰	Client Code
۱۳۰	استفاده از اشیا
۱۳۲	محافظت کردن (Encapsulation)
۱۳۵	خصوصیات (Properties)
۱۳۶	شکل کلی یک Property
۱۳۶	شرح ساختار Property
۱۳۸	استفاده از خصوصیات در کلاینت کد
۱۳۹	خصوصیات فقط نوشتنی (Write Only)
۱۴۰	خصوصیات فقط خواندنی
۱۴۰	پیاده سازی خودکار خصوصیات
۱۴۱	ارث بری از کلاس دیگر
۱۴۲	ساختار وارث (Inheritance hierarchy)
	نگاهی به کلاینت کد کلاس های پایه و کلاس های مشتق
۱۴۵	محدودیت دسترسی به اعضای کلاس
۱۴۵	Private
۱۴۶	Public
۱۴۶	Friend
۱۴۶	Protected
۱۴۹	Abstraction
۱۵۰	Sealed Class
۱۵۱	چند ریختی (PolyMorphism)
۱۵۴	متدهای Overridable
۱۵۵	متدهای Overloads
۱۵۵	متدهای Shadows
۱۵۵	متدهای Overrides
۱۵۶	NotOverridable
۱۵۷	MustOverride
۱۶۰	Dynamic Binding

۹۵	LINQ
۹۶	ساختار عمومی دستورات LINQ
۹۶	تشریح کلمات کلیدی این ساختار
۹۸	استفاده از شرط در دستورات LINQ
۹۹	Lambda In LINQ
۱۰۰	مرتب سازی مجموعه
۱۰۰	داده های غیر تکراری
۱۰۱	عملگر Select
	استفاده از عملگر Select با استفاده از روش فراخوانی
۱۰۲	متد
۱۰۲	عملگر SKIP
۱۰۳	عملگر Take
۱۰۷	مزایای استفاده از عملگر Select
۱۰۸	عملگر LIKE
۱۱۱	متد Any
۱۱۱	متد Contains
۱۱۲	متد Concat
۱۱۲	متد Union
۱۱۲	متد Intersect
۱۱۳	متد Except
۱۱۳	متد First, Last
	فصل چهارم : جهش به دنیای برنامه نویسی شی گرا
۱۱۵	(OOP)
۱۱۷	بررسی روند برنامه نویسی از گذشته تا کنون
	مفاهیم کلیدی OOP
۱۱۸	(Object Oriented Programming)
۱۱۸	شی (Object)
۱۲۱	کلاس (Class) چیست؟
۱۲۱	چه ارتباطی بین شی و کلاس وجود دارد؟
	شروع کدنویسی به سبک کلاس نویسی در VB.NET
۱۲۱	
۱۲۲	چندی از مزایای کلاس نویسی
۱۲۲	Server Code

استفاده از LINQ توسط متدهای الحاقی لیست

۲۰۸	
۲۰۸	Select متد
۲۰۹	Where متد
۲۱۰	ترکیب متدها
۲۱۰	ترکیب عملگر و متد
۲۱۱	عملگرهای مرتب سازی
۲۱۱	Reverse عملگر
۲۱۴	DefaultIfEmpty متد
۲۱۵	عملگرهای اتصال و دسته بندی
۲۱۵	Join عملگر
۲۱۷	گروه بندی Group By
۲۱۹	عملگر Group Join
۲۲۱	پایه سازی Left Outer Join
۲۲۱	Single متد
۲۲۳	توابع تراکمی
۲۲۴	SUM متد
۲۲۴	استفاده از توابع تراکمی در دستورات LINQ
۲۲۵	متد Count و LongCount
۲۲۶	متد Max و Min
۲۲۷	متد Average

فصل ششم: آشنایی با محیط Visual (تصویری) دات نت

۲۲۹	
۲۳۱	اصطلاح RAD
۲۳۱	اصطلاح IDE
۲۳۲	NET Framework
۲۳۴	کار با IDE
۲۳۵	شمای برنامه VS
۲۳۶	Snap Line
۲۳۷	دسترسی به کادر Application
۲۳۷	خلق اشیای بصری و شروع برنامه سازی
۲۳۷	چگونه می توان یک شی را ایجاد کرد و به آن دسترسی داشت ؟

۱۶۱	Contavariance
۱۶۲	Covariance
۱۶۳	Multiple Inheritance
۱۶۳	واسط (Interface)
۱۶۴	پایه سازی واسط
	تعریف عملکرد عملگرها در کلاسهای ایجاد شده
۱۶۹	
۱۷۱	ارتباطات (Relationships)
۱۷۱	ارتباط یک به یک
۱۷۶	ارتباطات یک به چند
۱۷۷	بخش بخش کردن کلاسها (Partial)
۱۷۸	متدهای الحاقی (Extension)
۱۸۰	مستند سازی کلاسها
۱۸۱	نمایش دیاگرام کلاسها
۱۸۳	ایجاد کلاس دیاگرام
۱۸۴	ایجاد خصوصیت، ویژگی و متد برای کلاسها
۱۸۴	ایجاد ویژگی
۱۸۴	ارتباطات
۱۸۵	وراثت گیری
۱۸۶	مستند سازی در کلاس دیاگرام
۱۸۸	فضای نامی (Namespace)
۱۹۰	عملگر With

تعریف کلاس با نوعهای بی نام (Anonymous Types)

۱۹۱	
۱۹۲	کتابخانه کلاسها (Class library)

فصل پنجم: کاربرد LINQ در اشیا

۱۹۹	
۲۰۱	LINQ TO OBJECTS
۲۰۱	شی List
۲۰۱	متد ADD
	افزودن مجموعه ای از عناصر هنگام تعریف لیست
۲۰۲	



تکنیک شکار رویدادها در زمان کدنویسی ۲۵۹	چگونه می‌توان به متدها و خصوصیات یک شی
کالبد شکافی رویداد ۲۶۱	دسترسی داشت؟ ۲۳۹
باقی ماندن تنظیمات اشیای فرم در بازگشت مجدد	از طریق پنجره Properties ۲۳۹
به برنامه ۲۶۵	از طریق SmartTag ۲۳۹
انتساب Setting به خصوصیات اشیا از طریق	مقایسه تغییر مقدار خصوصیت یک شی از دنیای
پنجره Properties ۲۶۷	واقعی با یک شی از دنیای VS ۲۴۰
خلق اشیای بصری دات نت در کدنویسی ۲۶۸	از طریق کدنویسی ۲۴۰
یک ترند ۲۷۴	بررسی خواص پرکاربرد اشیا دات نت ۲۴۱
ساختار Using ۲۷۵	خصوصیات با مقادیر عددی ۲۴۱
انتساب رویداد به شی در کدنویسی ۲۷۶	استفاده از Layout ۲۴۲
دستور Handles ۲۷۶	خصوصیات با مقادیر منطقی ۲۴۴
دستور AddHandler ۲۷۷	خصوصیات با مقادیر رشته‌ای ۲۴۴
فصل هفتم: کالبد شکافی پروژه، رویدادهای اشیا،	خصوصیات مربوط به رنگ اشیا ۲۴۴
چند فرمی‌ها ۲۷۹	خصوصیات Font ۲۴۵
فایل‌های مربوط به پروژه ۲۸۱	خصوصیت Tab Index و Tab Stop ۲۴۶
فرم چیست؟ ۲۸۲	تنظیم خصوصیت Tab Index اشیا با استفاده از
بررسی برخی از خواص و رویدادهای فرم ۲۸۲	ابزار Layout ۲۴۶
متدها ۲۸۴	متد Select ۲۴۷
رویدادهای اساسی مربوط به فرم ۲۸۴	خصوصیت Flat Style ۲۴۷
آنچه باید در رابطه با برنامه‌های چندفرمی بدانید!	خصوصیت Dock ۲۴۷
..... ۲۸۵	چگونگی ایجاد یک پروژه با VB.NET ۲۴۸
برنامه‌های MDI ۲۸۶	معرفی کنترل‌های پر کاربرد ۲۴۸
صفحات Splash ۲۸۸	کنترل Label ۲۴۸
کادر مکالمه About (درباره) ۲۹۰	کنترل Button ۲۴۹
طراحی فرم برای زبان‌های مختلف ۲۹۲	رویداد چیست؟ ۲۴۹
خصوصیت Language ۲۹۲	شرح برنامه نویسی ۲۵۱
تنظیمات ویژگی‌های پروژه ۲۹۴	Default Event ۲۵۲
فصل هشتم: معرفی کنترل‌های پیشرفته ۲۹۷	کنترل TextBox ۲۵۲
..... ۲۹۷	خصوصیات ۲۵۲
کنترل ToolTip ۲۹۹	متدها ۲۵۳
خصوصیات ۳۰۰	رویدادها ۲۵۳
کنترل Picture Box ۳۰۰	کنترل CheckBox ۲۵۶
	کنترل RadioButton ۲۵۷

۳۲۷	متدها
۳۲۹	کنترل LinkLabel
۳۲۹	خصوصیات
۳۲۹	رویداد
۳۴۱	کنترل Web Browser
۳۴۱	متدها
بررسی دو خصوصیت AutoCompleteMode و	
۳۴۱	AutoCompleteSource
۳۴۳	کنترل MaskedTextBox
۳۴۳	خصوصیات
۳۴۴	کنترل NumericUpDown
۳۴۴	خصوصیات
۳۴۵	کنترل DomainUpDown
۳۴۵	خصوصیات
۳۴۵	کنترل TrackBar
۳۴۵	خصوصیات

فصل نهم : تقسیم بندی فرم و گروه بندی اشیاء. ۳۴۹

۳۵۱	کنترل Group Box
۳۵۱	خصوصیات
۳۵۲	کنترل Panel
۳۵۲	خصوصیات
۳۵۳	نوار ابزار ToolBar
۳۵۴	کنترل ToolStrip
۳۵۵	کنترل TabControl
۳۵۷	خصوصیات
۳۵۹	کنترل StatusBar
۳۶۰	کنترل Splitter
۳۶۱	کنترل TableLayoutPanel
۳۶۲	کنترل SplitContainer
۳۶۳	کنترل FlowLayoutPanel
۳۶۳	کنترل ToolStripContainer

۳۰۲	خصوصیات
۳۰۲	کنترل Timer
۳۰۲	خصوصیات
۳۰۲	متدها
۳۰۴	کنترل ImageList
۳۰۴	خصوصیات
خصوصیت های ImageList و ImageIndex	
۳۰۵	و ImageAlign
۳۰۶	کنترل Menu Bar
۳۰۷	خصوصیات
۳۰۹	رویدادها
۳۱۰	کنترل ContextMenuStrip
۳۱۲	کنترل ListBox
۳۱۲	خصوصیات
۳۱۴	متدها
۳۱۴	رویداد
۳۱۶	کنترل CheckedListBox
۳۱۶	کنترل ComboBox
۳۱۶	خصوصیات
۳۱۷	رویداد
۳۲۲	کنترل DataGridView برای نمایش لیست
۳۲۴	کنترل Tree View
۳۲۵	خصوصیات TreeView
۳۲۶	متدهای TreeView
۳۲۷	خصوصیات TreeNode
۳۲۷	متدهای TreeNode
۳۲۹	کنترل ListView
۳۳۲	خصوصیات
۳۳۴	کنترل NotifyIcons
۳۳۴	خصوصیات
۳۳۶	رویدادها
۳۳۷	کنترل ProgressBar
۳۳۷	خصوصیات



استفاده از ساختار Try ... Catch... Finally...	فصل دهم : آشنایی با جعبه‌های متن چندخطی و
۳۹۸End Try Statement	۳۶۵.....کادرهای مکالمه
۴۰۱When دستور	۳۶۷.....متون چند خطی در کنترل TextBox
فصل دوازدهم : دوستانان را در VB.Net پیدا کنید!	۳۶۷.....کنترل RichTextBox
۴۰۳(IntelliSense,MY)	۳۶۷.....خصوصیات
۴۰۵معرفی IntelliSense به شما	۳۶۹.....متدها
۴۰۵در لیست IntelliSense چه چیزهایی قرار دارد؟..	۳۷۰.....آیا می‌دانید کادر مکالمه چیست؟
۴۰۵چگونه می‌توان از IntelliSense استفاده کرد؟..	۳۷۱.....ویژگی‌های کادرهای مکالمه
۴۰۶چگونه IntelliSense ظاهر می‌شود؟	۳۷۱.....MsgBox
۴۰۸تشخیص نوع متغیر توسط IntelliSense	۳۷۲.....InputBox
۴۰۸بر طرف کردن خطاهای زمان کدنویسی	۳۷۳.....کادرهای مکالمه از پیش تعریف شده
۴۱۰ایجاد با استفاده (Generate From Usage)	۳۷۳.....کنترل OpenFileDialog
۴۱۲My	۳۷۳.....متد ShowDialog
۴۱۲معرفی بخش‌های مختلف My	۳۷۳.....خصوصیات OpenFileDialog
فصل سیزدهم : آشنایی با Activexهای کوچک و	۳۷۴.....متد OpenFileDialog
۴۱۷پر کاربرد	۳۷۵.....کنترل SaveDialog
۴۱۹Activex چیست؟	۳۷۵.....خصوصیات SaveDialog
۴۱۹دیدگاه برنامه‌نویسان در مورد Activexها	۳۷۵.....متدهای SaveDialog
چگونه می‌توانید Activexها را در VB.NET به کار	۳۷۶.....کنترل FolderBrowserDialog
۴۱۹بگیرید؟	۳۷۶.....خصوصیات FolderBrowserDialog
۴۲۱معرفی Adobe PDF Reader	۳۷۷.....کنترل Colordialog
۴۲۲متدها	۳۷۷.....خصوصیات Colordialog
۴۲۴معرفی Shockwave Flash Object	۳۷۸.....کنترل FontDialog
۴۲۴خصوصیت Movie	۳۷۸.....خصوصیات FontDialog
۴۲۵متدها	۳۷۹.....کلاس InstalledFontCollection
۴۲۶استفاده از Agent در VB	۳۸۰.....DialogResult
۴۲۷متدها	فصل یازدهم : بررسی و مقابله با خطاها
۴۲۸اکتیوکس AnalogClock	۳۸۹.....مقدمه
۴۲۹خصوصیات	۳۹۰.....Error Checking چیست؟
۴۳۱ترفند زیبا سازی ساعت	۳۹۲.....شی ErrorProvider
۴۳۱تقویم فارسی (FarsiLibrary)	۳۹۷.....Error Handling چیست؟
۴۳۴ترفند فارسی کردن تقویم	۳۹۷.....استثنا (Exception)

۵۰۲.....	شی Command	۴۳۴.....	نرم افزار ارسال SMS
۵۰۵.....	فصل پانزدهم : برنامه نویسی افیس	۴۳۵.....	GSM مودم چیست؟
۵۰۷.....	ارتباط با نرم افزارهای Word.Excel	۴۳۵.....	پروتکل ارتباطی
۵۰۷.....	استفاده خودکار MSWORD	۴۳۷.....	mCore.net sms library
۵۰۸.....	معرفی کلاس‌های کاربردی نرم افزار MSWORD	۴۳۸.....	خصوصیات
۵۰۸.....	کلاس Application	۴۳۸.....	متدها
۵۰۹.....	ایجاد سند جدید در Word	۴۴۰.....	رویدادها
۵۱۰.....	باز کردن فایل در MSWORD	۴۴۵.....	فصل چهاردهم :آشنایی با DevComponents
۵۱۰.....	ذخیره کردن فایل در MSWORD	۴۴۷.....	معرفی DevComponents
۵۱۱.....	کلاس Document	۴۴۸.....	کنترل TextBoxX
۵۱۱.....	کلاس Range	۴۴۹.....	کنترل IntegerInput
۵۱۳.....	کلاس Words	۴۵۲.....	کنترل ReflectionImage
۵۱۴.....	ایجاد یک پاراگراف جدید در سند	۴۵۲.....	کنترل ReflectionLabel
۵۱۶.....	کلاس Paragraph	۴۵۳.....	کنترل Slider
۵۱۷.....	ایجاد جدول در سند	۴۵۴.....	کنترل ProgressBarX
۵۱۸.....	کلاس Table	۴۵۶.....	کنترل ButtonX
۵۱۸.....	تعیین رنگ جدول	۴۵۸.....	کنترل BalloonTip
۵۱۹.....	دسترسی به محدوده‌ی هر ستون	۴۶۲.....	کنترل SuperTooltip
۵۲۰.....	افزودن عکس به سند	۴۶۴.....	کنترل ComboBoxEx
۵۲۰.....	کلاس Shape	۴۶۶.....	کنترل AdvTree
۵۲۱.....	استفاده خودکار EXCEL	۴۷۱.....	کنترل BubbleBar
۵۲۱.....	معرفی کلاس‌های نرم افزار EXCEL	۴۷۳.....	کنترل TabStrip
۵۲۱.....	کلاس Application	۴۷۳.....	کنترل ContextMenuBar
۵۲۱.....	ایجاد Workbook جدید در Excel	۴۷۵.....	کنترل PanelEx
۵۲۳.....	باز کردن فایل Excel	۴۷۶.....	کنترل GroupPanel
۵۲۳.....	ذخیره‌ی فایل Excel	۴۷۷.....	کنترل Expandable Panel
۵۲۴.....	ایجاد Sheet جدید	۴۸۰.....	کنترل ExplorerBar
۵۲۴.....	دسترسی به سلولها	۴۸۳.....	کنترل Itempanel
۵۲۴.....	خصوصیت Cells	۴۸۵.....	کنترل SideBar
۵۲۵.....	کلاس Range	۴۸۷.....	کنترل NavigationPane
		۴۹۱.....	کنترل DotNetBarManager
		۴۹۴.....	کنترل RibbonControl



۵۶۲.....ارتباطات	فصل شانزدهم: آشنایی با فایل‌ها.....۵۲۷
۵۶۳.....ارتباط یک به یک (۱:۱)	فایل‌ها.....۵۲۹
۵۶۵.....محدودیت جامعیت ارجاعی چیست (RIR) ؟	خواندن و نوشتن اطلاعات.....۵۳۰

۶۲۳.....	شی SqlDataReader	۵۸۵.....	ایجاد View
۶۲۴.....	متد Read	۵۸۸.....	دستور Exists
۶۲۵.....	شی DataSet	۵۸۹.....	روال های ذخیره شده
۶۲۶.....	شی DataAdapter	فصل هجدهم : معماری برنامه های کاربردی بانک	
۶۲۷	وارد و ایجاد نمودن شی OleDbDataAdapter	۵۹۳.....	اطلاعاتی (به روش سنتی)
	همکاری شی OleDbDataAdapter با اشیا	۵۹۵.....	بانک های اطلاعاتی
۶۲۹.....	Data Command	۵۹۵.....	دسترسی به داده ها
۶۲۹.....	SelectCommand خصوصیت		دسترسی به بانک های اطلاعاتی در VB.NET2010
۶۳۱.....	اشیا Data aware	۵۹۶.....	
۶۳۳.....	استفاده از شی DataGrid	۵۹۶.....	معرفی بانک اطلاعاتی به VB.NET
	بروزرسانی داده ها توسط شی OleDbDataAdapter	۵۹۹.....	اتصال به SQL Server
۶۳۸.....		۶۰۳.....	ایجاد یک ارتباط ساده
۶۴۰.....	ساخت یک پارامتر	۶۰۳.....	شی Connection
	ایجاد و پیکربندی شی SqlDataAdapter در		انجام مرحله اتصال به منبع داده توسط کدنویسی
۶۴۴.....	کدنویسی	۶۰۵.....	
	مقداردهی خصوصیات SqlDataAdapter برای انجام	۶۰۶.....	خصوصیات
۶۴۵.....	فعالیت های CRUD	۶۰۶.....	متدها
۶۴۶.....	خصوصیت SelectCommand	۶۰۷.....	شی Data Command
۶۴۷.....	خصوصیت UpdateCommand	۶۰۷.....	ایجاد شی Data Command
۶۴۹.....	خصوصیت InsertCommand		انتساب شی Connection به شی DataCommand
۶۵۱.....	خصوصیت DeleteCommand	۶۰۸.....	
۶۵۲.....	ایجاد شی دیتاست، ساختاردهی و مقداردهی آن	۶۰۸.....	خصوصیت Connection
	ذخیره ی داده های درون DataSet در فایل XML و		تعیین دستورات SQL برای شی Data Command
۶۵۲.....	باز یابی آن	۶۰۸.....	
۶۵۲.....	متد WriteXml		خصوصیات CommandType و CommandText
۶۵۳.....	متد ReadXml	۶۰۹.....	
۶۵۴.....	جادوگر OleDbDataAdapter	۶۱۳.....	استفاده از پارامترها
	نحوه انجام عملیات جادوگر OleDbDataAdapter	۶۱۴.....	خصوصیت Parametrs
۶۵۴.....			ایجاد و استفاده از DataCommand در کدنویسی
	فصل نوزدهم: دسترسی به بانک اطلاعاتی با استفاده	۶۱۵.....	(زمان اجرا)
۶۶۱.....	از DataSet و TableAdapter	۶۱۷.....	استفاده از پارامترها در کدنویسی
	مدل سازی بانک به صورت دستی در Typed DataSet	۶۱۸.....	متد ExecuteNonQuery
۶۶۳.....		۶۲۰.....	متد ExecuteScalar



۶۹۴	مثال جامع CRUD	مدل سازی جداول با استفاده از پنجره‌ی Server Explorer
۶۹۵	LINQ To DataSet	۶۶۵
۶۹۵	استخراج داده‌ها توسط LINQ	۶۶۶
۷۰۰	تکنیک پر کردن دیتاست بوسیله‌ی LINQ	۶۶۹
	مقید سازی داده‌ها با استفاده از BindingSource	۶۷۰
۷۰۱		۶۷۰
۷۰۱	اتصال DataSet به BindingSource	۶۷۱
۷۰۲	در زمان طراحی	۶۷۳
۷۰۳	در زمان اجرا	۶۷۴
۷۰۳	اتصال منبع اشیاء به BindingSource	۶۷۴
	ارتباط کنترل‌های ویندوز فرم به BindingSource	نحوه‌ی کدنویسی (در ویرایشگر کد) برای خصوصیت
۷۰۴		۶۷۵
	ایجاد یک ارتباط ساده بین BindingSource و	۶۷۵
۷۰۵	TextBox	نمونه‌گیری از جداول درون دیتاست
	ارتباط مرکب با اشیاء ListBox و ComboBox	۶۷۶
۷۰۷		۶۷۷
۷۰۷	خصوصیت DataSource	استفاده از TableAdapter در کدنویسی
۷۰۷	خصوصیت DisplayMember	۶۷۸
۷۰۸	خصوصیت SelectedValue, ValueMember	۶۷۸
	کدنویسی ارتباط BindingSource و کنترل‌های	۶۷۹
۷۰۹	ListBox و ComboBox	۶۷۹
۷۱۰	ارتباط BindingSource با DataGridView	۶۸۰
	بررسی خصوصیات و متدهای شیء BindingSource	۶۸۰
۷۱۱		۶۸۱
۷۱۳	مثال جامع	۶۸۲
	نمایش روابط با استفاده از کنترل BindingSource	۶۸۲
۷۱۴		۶۸۳
۷۱۴	BindingSource فرزند	بدست آوردن محل اشاره گر و حرکت بین رکوردها
۷۱۷	کنترل DataView	۶۹۲
۷۱۸	حذف اطلاعات توسط شیء DataView	انتقال اشاره گر به رکورد بعدی
۷۱۸	خصوصیات	۶۹۲
۷۲۱	ذخیره‌ی جداول بانک در فایل اکسل	۶۹۳
۷۲۲	ذخیره‌ی جداول بانک در فایل Word	۶۹۳
		رفتار به رکورد آخر
		رفتار به رکورد اول
		لغو تغییرات

۷۶۱	 DataBase (دیتابیس)
۷۶۱	 Entity Data Model
۷۶۱	 بخش منطق (Logical)
۷۶۱	 بخش (Conceptual)
۷۶۴	 Mapping بخش
۷۶۴	 لایه‌های Entity SQL و Entity Client
۷۶۴	 لایه‌های LINQ to Entities و Object Services
۷۶۵	 ایجاد اولین مدل از روی پایگاه داده‌ها
	EDM(Entity Data Model) اعتبار سنجی مدلها در
۷۶۹	
۷۷۰	 Mapping Details پنجره‌ی
۷۷۲	 Navigation Properties
۷۷۲	 اجرای پرس و جو بر روی مدل
۷۷۳	 اجرای پرس و جو بر روی جدول والد و فرزند
۷۷۴	 درج اطلاعات
۷۷۴	 درج اطلاعات در جدول پدر و فرزند
۷۷۵	 درج اطلاعات در جدول فرزند و پدر
۷۷۵	 حذف اطلاعات
۷۷۶	 ویرایش اطلاعات
	ارتباطات چند به چند (M:N) یا Many to Many(*:*)
۷۷۷	
۷۸۴	 درج اطلاعات در موجودیت فصل مشترک
۷۸۴	 پرس و جو از موجودیت فصل مشترک
۷۸۵	 حذف اطلاعات از موجودیت فصل مشترک
۷۸۷	 استفاده از View
۷۸۸	 ارتباط مدل EDM با کنترل‌های فرم
۷۹۱	 Entity قرار دادن دو جدول در یک
۷۹۶	 نگاهی به کد مدل منطقی (SSDL)
۷۹۷	 نگاهی به کد لایه ادراکی
۷۹۷	 مقایسه کدهای لایه منطقی و ادراکی
۷۹۸	 نگاهی به کدهای مدل نقشه کشی (MSL)
۷۹۹	 Condition

فصل بیستم : LINQ TO SQL ۷۲۵

۷۲۸	 یک مثال کوچک از ORM
۷۳۱	 LINQ TO SQL (L2S)
۷۳۲	 ایجاد ORM استاندارد L2S
۷۳۶	 DataContext
۷۳۸	 متد GetCommand
۷۳۸	 اضافه کردن داده‌ها
۷۴۰	 حذف داده‌ها
۷۴۱	 ویرایش داده‌ها
۷۴۲	 LOG خصوصیت
۷۴۳	 CRUD مثال جامع
	ایجاد DataContext و جداول کلاس‌های نماینده به
۷۴۵	 کمک ابزارهای طراحی
	مدل سازی جداول با استفاده از پنجره‌ی
۷۴۶	 Server Explorer
۷۴۸	 اضافه کردن یک متد جدید به DataContext
	اجرای کدهای دلخواه برای متدهای CUD مربوط به
۷۴۸	 شی DataContext
۷۵۰	 ایجاد ارتباط بین جداول کلاس‌های نماینده
۷۵۱	 اجرای پرس و جو بر روی کلاس پدر و فرزند
	اجرای پرس و جو بر روی جدول پدر و جدول فرزند
۷۵۲	 درون پدر
	اجرای پرس و جو بر روی جدول فرزند و جدول پدر
۷۵۳	 درون فرزند
	اضافه کردن ردیف‌های فرزند توسط جدول پدر
۷۵۵	

فصل بیست و یکم : ADO.NET EntityFramework ۷۵۷

۷۵۷	 چرا مایکروسافت دو تکنولوژی مختلف ORM تولید کرده است؟
۷۵۹	 Entity SQL چیست؟
۷۶۰	 معماری ADO.net Entity Framework



۸۳۷	شی EntityCommand
۸۳۷	شی EntityDataReader
۸۳۸	استفاده از پارامتر
فصل بیست و دوم : آشنایی با برنامه نویسی سه لایه	
۸۴۱	
۸۴۳	برنامه نویسی سه لایه
۸۴۳	تشریح لایه‌ها
۸۴۷	Business Entity Layer
۸۴۹	کتابخانه‌ی Data Access Layer
۸۴۹	کتابخانه‌ی Business Logic Layer
۸۴۹	کتابخانه‌ی Business Entity
۸۵۰	پروژه Interface
	استفاده از تکنولوژی‌های جدید در برنامه نویسی سه
۸۵۹	لایه
۸۶۰	ایجاد لایه‌ی Data LINQ Layer
فصل بیست و سوم : گزارش و تولید آن	
۸۶۹	گزارش
۸۶۹	Crystal Reports
۸۶۹	ایجاد یک گزارش ساده
۸۷۴	معرفی محیط گزارش
۸۷۵	Report Header
۸۷۶	Page Header
۸۷۷	Details
۸۷۷	Report Footer
۸۷۸	Page Footer
۸۸۰	وضعیت‌های یک گزارش
۸۸۰	قراردادن فیلدها در قسمت Details
۸۸۲	خلق اشیا و قرار دادن آنها در بخش‌های مختلف
۸۸۳	شی Line
۸۸۴	تعیین فرمت خط
۸۸۵	شی Box

پرس و جو بر روی موجودیتی دارای Condition	۸۰۱
وراثت	۸۰۲
مدیریت یک جدول توسط چند موجودیت بوسیله‌ی وراثت (Table-per- hierarchy)	۸۰۲
ورود داده‌ها در جدول توسط موجودیت نوع جدید (وراثت گرفته شده TPH)	۸۰۹
اجرای پرس و جو بر روی جدول توسط موجودیت نوع جدید (وراثت گرفته شده TPH)	۸۱۰
حذف داده‌ها از جدول توسط موجودیت نوع جدید (وراثت گرفته شده TPH)	۸۱۱
ویرایش اطلاعات جدول توسط موجودیت نوع جدید (وراثت گرفته شده TPH)	۸۱۱
مدیریت چند جدول توسط یک موجودیت بوسیله‌ی وراثت (Table-per- Type)	۸۱۳
درج اطلاعات در جداول توسط موجودیت نوع جدید (وراثت گرفته شده TPT)	۸۱۶
اجرای پرس و جو بر روی جدول توسط موجودیت نوع جدید (وراثت گرفته شده TPT)	۸۱۶
حذف اطلاعات از جداول توسط موجودیت نوع جدید (وراثت گرفته شده TPT)	۸۱۷
ویرایش اطلاعات از جداول توسط موجودیت نوع جدید (وراثت گرفته شده TPT)	۸۱۷
تعیین کدهای CUD	۸۱۸
مدل سازی بدون پایگاه داده‌ها	۸۲۲
ایجاد یک موجودیت جدید	۸۲۴
ایجاد خصوصیات جدید	۸۲۵
ایجاد ارتباط بین موجودیت‌ها	۸۲۷
خصوصیات Complex (مرکب)	۸۳۱
کتابخانه‌ی Entity Client	۸۳۴
آشنایی مختصر با زبان eSQL	۸۳۵
آشنائی با کلاس‌های Entity Client	۸۳۶
شی EntityConnection	۸۳۶

۶۲۳.....	شی SqlDataReader	۵۸۵.....	ایجاد View
۶۲۴.....	متد Read	۵۸۸.....	دستور Exists
۶۲۵.....	شی DataSet	۵۸۹.....	روال های ذخیره شده
۶۲۶.....	شی DataAdapter	فصل هجدهم: معماری برنامه های کاربردی بانک	
۶۲۷.....	وارد و ایجاد نمودن شی OleDbDataAdapter	۵۹۳.....	اطلاعاتی (به روش سنتی)
	همکاری شی OleDbDataAdapter با اشیا	۵۹۵.....	بانک های اطلاعاتی
۶۲۹.....	Data Command	۵۹۵.....	دسترسی به داده ها
۶۲۹.....	SelectCommand خصوصیت		دسترسی به بانک های اطلاعاتی در VB.NET2010
۶۳۱.....	اشیا Data aware	۵۹۶.....	
۶۳۳.....	استفاده از شی DataGrid	۵۹۶.....	معرفی بانک اطلاعاتی به VB.NET
	بروزرسانی داده ها توسط شی OleDbDataAdapter	۵۹۹.....	اتصال به SQL Server
۶۳۸.....		۶۰۳.....	ایجاد یک ارتباط ساده
۶۴۰.....	ساخت یک پارامتر	۶۰۳.....	شی Connection
	ایجاد و پیکربندی شی SqlDataAdapter در		انجام مرحله اتصال به منبع داده توسط کدنویسی
۶۴۴.....	کدنویسی	۶۰۵.....	
	مقداردهی خصوصیات SqlDataAdapter برای انجام	۶۰۶.....	خصوصیات
۶۴۵.....	فعالیت های CRUD	۶۰۶.....	متدها
۶۴۶.....	خصوصیت SelectCommand	۶۰۷.....	شی Data Command
۶۴۷.....	خصوصیت UpdateCommand	۶۰۷.....	ایجاد شی Data Command
۶۴۹.....	خصوصیت InsertCommand		انتساب شی Connection به شی DataCommand
۶۵۱.....	خصوصیت DeleteCommand	۶۰۸.....	
۶۵۲.....	ایجاد شی دیتاست، ساختاردهی و مقداردهی آن	۶۰۸.....	خصوصیت Connection
	ذخیره ی داده های درون DataSet در فایل XML و		تعیین دستورات SQL برای شی Data Command
۶۵۲.....	بازیابی آن	۶۰۸.....	
۶۵۲.....	متد WriteXml		خصوصیات CommandType و CommandText
۶۵۳.....	متد ReadXml	۶۰۹.....	
۶۵۴.....	جادوگر OleDbDataAdapter	۶۱۳.....	استفاده از پارامترها
	نحوه انجام عملیات جادوگر OleDbDataAdapter	۶۱۴.....	خصوصیت Parametrs
۶۵۴.....			ایجاد و استفاده از DataCommand در کدنویسی
	فصل نوزدهم: دسترسی به بانک اطلاعاتی با استفاده	۶۱۵.....	(زمان اجرا)
۶۶۱.....	از DataSet و TableAdapter	۶۱۷.....	استفاده از پارامترها در کدنویسی
	مدل سازی بانک به صورت دستی در Typed DataSet	۶۱۸.....	متد ExecuteNonQuery
۶۶۳.....		۶۲۰.....	متد ExecuteScalar



۶۹۴	مثال جامع CRUD	مدل سازی جداول با استفاده از پنجره‌ی Server Explorer
۶۹۵	LINQ To DataSet	۶۶۵
۶۹۵	استخراج داده‌ها توسط LINQ	۶۶۶
۷۰۰	تکنیک پر کردن دیتاست بوسیله‌ی LINQ	۶۶۹
	BindingSource از استفاده از	۶۷۰
۷۰۱		۶۷۰
۷۰۱	اتصال DataSet به BindingSource	۶۷۱
۷۰۲	در زمان طراحی	۶۷۳
۷۰۳	در زمان اجرا	۶۷۴
۷۰۳	اتصال منبع اشیاء به BindingSource	۶۷۴
	ارتباط کنترل‌های ویندوز فرم به BindingSource	نحوه‌ی کدنویسی (در ویرایشگر کد) برای خصوصیت
۷۰۴		۶۷۵
	ایجاد یک ارتباط ساده بین BindingSource و	۶۷۵
۷۰۵	TextBox	۶۷۶
	ارتباط مرکب با اشیاء ListBox و ComboBox	۶۷۷
۷۰۷		۶۷۸
۷۰۷	خصوصیت DataSource	۶۷۸
۷۰۷	خصوصیت DisplayMember	۶۷۹
۷۰۸	خصوصیت SelectedValue, ValueMember	۶۷۹
	کدنویسی ارتباط BindingSource و کنترل‌های	۶۷۹
۷۰۹	ListBox و ComboBox	۶۸۰
۷۱۰	ارتباط BindingSource با DataGridView	۶۸۰
	بررسی خصوصیات و متدهای شیء BindingSource	۶۸۱
۷۱۱		۶۸۲
۷۱۳	مثال جامع	۶۸۲
	نمایش روابط با استفاده از کنترل BindingSource	۶۸۳
۷۱۴		بدست آوردن محل اشاره گر و حرکت بین رکوردها
۷۱۴	BindingSource فرزند	۶۹۲
۷۱۷	کنترل DataView	۶۹۲
۷۱۸	حذف اطلاعات توسط شیء DataView	۶۹۲
۷۱۸	خصوصیات	۶۹۳
۷۲۱	ذخیره‌ی جداول بانک در فایل اکسل	۶۹۳
۷۲۲	Word ذخیره‌ی جداول بانک در فایل	۶۹۳
		لغو تغییرات

۷۶۱	 DataBase (دیتابیس)
۷۶۱	 Entity Data Model
۷۶۱	 بخش منطق (Logical)
۷۶۱	 بخش (Conceptual)
۷۶۴	 بخش Mapping
۷۶۴	 لایه‌های Entity Client و Entity SQL
۷۶۴	 لایه‌های LINQ to Entities و Object Services
۷۶۵	 ایجاد اولین مدل از روی پایگاه داده‌ها
	EDM(Entity Data Model) اعتبار سنجی مدلها در
۷۶۹	
۷۷۰	 Mapping Details پنجره‌ی
۷۷۲	 Navigation Properties
۷۷۲	 اجرای پرس و جو بر روی مدل
۷۷۳	 اجرای پرس و جو بر روی جدول والد و فرزند
۷۷۴	 درج اطلاعات
۷۷۴	 درج اطلاعات در جدول پدر و فرزند
۷۷۵	 درج اطلاعات در جدول فرزند و پدر
۷۷۵	 حذف اطلاعات
۷۷۶	 ویرایش اطلاعات
	ارتباطات چند به چند (M:N) یا Many to Many(*:*)
۷۷۷	
۷۸۴	 درج اطلاعات در موجودیت فصل مشترک
۷۸۴	 پرس و جو از موجودیت فصل مشترک
۷۸۵	 حذف اطلاعات از موجودیت فصل مشترک
۷۸۷	 استفاده از View
۷۸۸	 ارتباط مدل EDM با کنترل‌های فرم
۷۹۱	 قراردادن دو جدول در یک Entity
۷۹۶	 نگاهی به کد مدل منطقی (SSDL)
۷۹۷	 نگاهی به کد لایه ادراکی
۷۹۷	 مقایسه کدهای لایه منطقی و ادراکی
۷۹۸	 نگاهی به کدهای مدل نقشه کشی (MSL)
۷۹۹	 Condition

فصل بیستم : LINQ TO SQL

۷۲۸	 یک مثال کوچک از ORM
۷۳۱	 LINQ TO SQL(L2S)
۷۳۲	 ایجاد ORM استاندارد L2S
۷۳۶	 DataContext
۷۳۸	 متد GetCommand
۷۳۸	 اضافه کردن داده‌ها
۷۴۰	 حذف داده‌ها
۷۴۱	 ویرایش داده‌ها
۷۴۲	 LOG خصوصیت
۷۴۳	 مثال جامع CRUD
	ایجاد DataContext و جداول کلاس‌های نماینده به
۷۴۵	 کمک ابزارهای طراحی
	مدل سازی جداول با استفاده از پنجره‌ی
۷۴۶	 Server Explorer
۷۴۸	 اضافه کردن یک متد جدید به DataContext
	اجرای کدهای دلخواه برای متدهای CUD مربوط به
۷۴۸	 شی DataContext
۷۵۰	 ایجاد ارتباط بین جداول کلاس‌های نماینده
۷۵۱	 اجرای پرس و جو بر روی کلاس پدر و فرزند
	اجرای پرس و جو بر روی جدول پدر و جدول فرزند
۷۵۲	 درون پدر
	اجرای پرس و جو بر روی جدول فرزند و جدول پدر
۷۵۳	 درون فرزند
	اضافه کردن ردیف‌های فرزند توسط جدول پدر
۷۵۵	

فصل بیست و یکم : ADO.NET EntityFramework

۷۵۷	
	چرا مایکروسافت دو تکنولوژی مختلف ORM تولید
۷۵۹	 کرده است؟
۷۶۰	 Entity SQL چیست؟
۷۶۰	 معماری ADO.net Entity Framework



۸۳۷	شی EntityCommand
۸۳۷	شی EntityDataReader
۸۳۸	استفاده از پارامتر
فصل بیست و دوم : آشنایی با برنامه نویسی سه لایه	
۸۴۱	
۸۴۳	برنامه نویسی سه لایه
۸۴۳	تشریح لایه‌ها
۸۴۷	Business Entity Layer
۸۴۹	کتابخانه‌ی Data Access Layer
۸۴۹	کتابخانه‌ی Business Logic Layer
۸۴۹	کتابخانه‌ی Business Entity
۸۵۰	پروژه Interface
	استفاده از تکنولوژی‌های جدید در برنامه نویسی سه
۸۵۹	لایه
۸۶۰	ایجاد لایه‌ی Data LINQ Layer
فصل بیست و سوم : گزارش و تولید آن	
۸۶۹	گزارش
۸۶۹	Crystal Reports
۸۶۹	ایجاد یک گزارش ساده
۸۷۴	معرفی محیط گزارش
۸۷۵	Report Header
۸۷۶	Page Header
۸۷۷	Details
۸۷۷	Report Footer
۸۷۸	Page Footer
۸۸۰	وضعیت‌های یک گزارش
۸۸۰	قراردادن فیلدها در قسمت Details
۸۸۲	خلق اشیا و قرار دادن آنها در بخش‌های مختلف
۸۸۳	شی Line
۸۸۴	تعیین فرمت خط
۸۸۵	شی Box

پرس و جو بر روی موجودیتی دارای Condition	۸۰۱
وراثت	۸۰۲
مدیریت یک جدول توسط چند موجودیت بوسیله‌ی وراثت (Table-per- hierarchy)	۸۰۲
ورود داده‌ها در جدول توسط موجودیت نوع جدید (وراثت گرفته شده TPH)	۸۰۹
اجرای پرس و جو بر روی جدول توسط موجودیت نوع جدید (وراثت گرفته شده TPH)	۸۱۰
حذف داده‌ها از جدول توسط موجودیت نوع جدید (وراثت گرفته شده TPH)	۸۱۱
ویرایش اطلاعات جدول توسط موجودیت نوع جدید (وراثت گرفته شده TPH)	۸۱۱
مدیریت چند جدول توسط یک موجودیت بوسیله‌ی وراثت (Table-per- Type)	۸۱۳
درج اطلاعات در جداول توسط موجودیت نوع جدید (وراثت گرفته شده TPT)	۸۱۶
اجرای پرس و جو بر روی جدول توسط موجودیت نوع جدید (وراثت گرفته شده TPT)	۸۱۶
حذف اطلاعات از جداول توسط موجودیت نوع جدید (وراثت گرفته شده TPT)	۸۱۷
ویرایش اطلاعات از جداول توسط موجودیت نوع جدید (وراثت گرفته شده TPT)	۸۱۷
تعیین کدهای CUD	۸۱۸
مدل سازی بدون پایگاه داده‌ها	۸۲۲
ایجاد یک موجودیت جدید	۸۲۴
ایجاد خصوصیات جدید	۸۲۵
ایجاد ارتباط بین موجودیت‌ها	۸۲۷
خصوصیات Complex (مرکب)	۸۳۱
کتابخانه‌ی Entity Client	۸۳۴
آشنایی مختصر با زبان eSQL	۸۳۵
آشنائی با کلاس‌های Entity Client	۸۳۶
شی EntityConnection	۸۳۶

۹۲۹	مزایای HTML نسبت به RTF
۹۲۹	نرم افزار WinCHM
۹۳۳	قسمت ویرایش (Edit)
۹۳۴	قسمت سورس (Source)
۹۳۴	قسمت نمایش (Preview)
۹۳۵	شروع ساخت یک صفحه‌ی راهنما
۹۳۵	ایجاد لینک
۹۳۶	وارد کردن عکس
۹۳۶	ایجاد یک جدول
۹۳۷	تولید فایل برنامه‌ی راهنما
	استفاده از فایل‌های CHM در پروژه‌های NET
۹۳۸	
۹۳۸	کنترل HelpProvider
۹۳۸	نمایش پنجره‌ی راهنما برای شی مورد نظر
۹۴۳	ضمیمه الف (A): مراجع توابع دات نت
۹۴۵	توابع
۹۴۶	توابع ریاضیاتی (Math)
۹۴۶	Abs()
۹۴۶	Atan()
۹۴۶	Cos()
۹۴۷	Sin()
۹۴۷	Tan()
۹۴۷	Exp()
۹۴۷	Int() , Fix()
۹۴۹	Round()
۹۴۹	Log()
۹۴۹	Hex() , Oct()
۹۵۰	Pow()
۹۵۰	Sqrt()
۹۵۰	توابع رشته‌ای
۹۵۰	Asc() , AscW()
۹۵۱	Chr() , ChrW()

۸۸۶	شی Picture
۸۸۶	شی Text Object
۸۸۶	فیلدهای مخصوص
۸۸۶	معرفی فیلدهای مخصوص
۸۸۷	فیلدهای فرمولی
۸۸۹	پارامترها
	نمایش رکوردهای مورد نظر در زمان اجرای گزارش
۸۹۱	
۸۹۲	تعیین نوع فرمت فیلدها
۸۹۴	نمایش اطلاعات خاص با فرمت متفاوت
۸۹۵	استفاده از جداول پدر و فرزند در گزارشات
۸۹۶	ایجاد گروه
۹۰۰	فیلد Summray
۹۰۲	ایجاد زیرگزارش (Sub Report)
۹۱۲	انواع شرایط
۹۱۴	معرفی توابع پرکاربرد
۹۱۵	فراخوانی گزارش‌ها در پروژه
۹۱۶	ایجاد شی گزارش در کدنویسی
۹۱۶	تعیین دیتاست برای شی گزارش
۹۱۸	استفاده از زبان LINQ در گزارشات
۹۱۹	متد Load
۹۱۹	تغییر اتصال در زمان اجرا
	تنظیمات نمایشی کنترل Crystal Report Viewer
۹۲۰	
	متدهای مربوط به شی CrystalReportViewer
۹۲۱	
۹۲۲	مقداردهی پارامترها در خارج از گزارش
	مقداردهی شی Text خارج از گزارش (در کدنویسی)
۹۲۴	
	فصل بیست و چهارم: تولید برنامه‌های راهنما
۹۲۹	برنامه‌ی ساخت راهنما
۹۲۹	فایل CHM



۹۶۶.....DateAdd()	۹۵۱.....LCase() , Ucase()
۹۶۷.....DateDiff()	InStr([startPos,] string1,string2 [,compare])
۹۶۸.....DatePart()	۹۵۲.....
۹۶۸.....تاریخ شمسی	StrComp (string1,string2 [,start][,compare])
۹۶۹.....توابع قالب‌دهی داده‌ها	۹۵۳.....
۹۶۹.....Format()	۹۵۴.....Left()
۹۶۹.....کاراکترهای قالب‌دهی تاریخ و زمان	۹۵۴.....Right()
۹۷۲.....FormatCurrency()	۹۵۴.....Mid()
۹۷۲.....FormatDateTime()	۹۵۵.....Mid()=new_string
۹۷۳.....FormatNumber()	۹۵۵.....Len()
۹۷۴.....FormatPercent()	۹۵۶.....LTrim(),RTrim(),Trim()
۹۷۴.....LSet() , RSet()	۹۵۷.....Space()
۹۷۴.....Val() , Str()	۹۵۷.....StrDup()
۹۷۵.....توابع تبدیل	۹۵۸.....StrConv()
۹۷۶.....CType()	۹۵۹.....StrReverse()
(Data Type Identification) توابع شناسایی نوع داده	۹۵۹.....Replace()
۹۷۶.....	۹۶۰.....Join()
۹۷۷.....IsArray()	۹۶۰.....Split()
۹۷۷.....IsDate()	۹۶۱.....توابع تاریخ و ساعت
۹۷۷.....IsDBNull()	۹۶۱.....Now()
۹۷۷.....IsNothing()	۹۶۲.....Day()
۹۷۸.....IsNumeric()	۹۶۲.....Weekday()
۹۷۸.....IsReference()	۹۶۳.....WeekdayName()
۹۷۸.....TypeName()	۹۶۳.....Month ()
۹۷۹.....VarType()	۹۶۴.....MonthName()
۹۸۰.....توابع اعداد تصادفی	۹۶۴.....Year ()
۹۸۰.....Rnd()	۹۶۴.....Hour()
۹۸۱.....ضمیمه ب (B) : پروژه	۹۶۴.....Minute()
۹۸۳.....معرفی پروژه	۹۶۴.....Second()
۹۸۴.....عناصر پروژه	۹۶۵.....DateSerial()
۹۸۴.....بانک اطلاعاتی	۹۶۵.....DateValue()
۹۸۴.....جدول اعضا (TBL_Member)	۹۶۵.....TimeSerial()
	۹۶۶.....TimeValue()

۱۰۱۸.....AmanatDAL	شرح متدهای کلاس	۹۸۷.....	جدول کتاب (TBL_Book)
۱۰۱۹.....UserDAL	کلاس	۹۸۹.....	جدول امانات (TBL_Amanat)
۱۰۱۹.....UserDAL	شرح فیلدهای کلاس		ارتباط بین جداول TBL_Book, TBL_Member و TBL_Amanat
۱۰۱۹.....UserDAL	شرح متدهای کلاس	۹۹۱.....	جدول کاربر (TBL_User)
۱۰۲۰.....MailDAL	کلاس	۹۹۴.....	جدول پیام (TBL_Mail)
۱۰۲۰.....MailDAL	شرح فیلدهای کلاس		ارتباط بین جداول TBL_Mail و TBL_User
۱۰۲۱.....MailDAL	شرح متدهای کلاس	۹۹۹.....	پایه سازی لایه‌های پروژه
پایه سازی لایه‌ی (BAL) Business Access Layer		۱۰۰۰.....	پایه سازی لایه Business Entity
۱۰۲۲.....		۱۰۰۱.....	کتابخانه Business Entity
۱۰۲۲.....BAL	کتابخانه‌ی	۱۰۰۲.....	کلاس EntAmanat
۱۰۲۳.....MemberBAL	کلاس	۱۰۰۳.....	کلاس EntBook
۱۰۲۳.....MemberBAL	شرح فیلدهای کلاس	۱۰۰۳.....	کلاس EntMember
۱۰۲۳.....MemberBAL	شرح متدهای کلاس	۱۰۰۳.....	کلاس EntUser
۱۰۲۶.....BookBAL	کلاس	۱۰۰۴.....	کلاس EntMail
۱۰۲۶.....BookBAL	شرح فیلدهای کلاس	۱۰۰۴.....	پایه سازی لایه Data Access
۱۰۲۶.....BookBAL	شرح متدهای کلاس	۱۰۰۴.....	کتابخانه DAL
۱۰۳۰.....AmanatBAL	کلاس	۱۰۰۵.....	ایجاد DataSet1
۱۰۳۰.....AmanatBAL	شرح فیلدهای کلاس	۱۰۰۶.....	TBL_MemberDataTable
۱۰۳۰.....AmanatBAL	شرح متدهای کلاس	۱۰۱۰.....	TBL_BookDataTable
۱۰۳۳.....UserBAL	کلاس	۱۰۱۰.....	TBL_AmanatDataTable
۱۰۳۳.....UserBAL	شرح فیلدهای کلاس	۱۰۱۱.....	TBL_UserDataTable
۱۰۳۵.....UserBAL	شرح متدهای کلاس	۱۰۱۱.....	TBL_MailDataTable
۱۰۴۱.....MailBAL	کلاس	۱۰۱۲.....	RptAmanatMemberTableAdapter
۱۰۴۱.....MailBAL	شرح فیلدهای کلاس	۱۰۱۴.....	کلاس MemberDAL
۱۰۴۲.....MailBAL	شرح متدهای کلاس	۱۰۱۴.....	شرح فیلدهای کلاس MemberDAL
۱۰۴۳.....Interface	پایه سازی لایه	۱۰۱۴.....	شرح متدهای کلاس MemberDAL
۱۰۴۳.....Presentation	کتابخانه	۱۰۱۵.....	کلاس BookDAL
۱۰۴۵.....	فرم ثبت اعضا	۱۰۱۵.....	شرح فیلدهای کلاس BookDAL
۱۰۴۵.....	طراحی فرم ثبت اعضا	۱۰۱۶.....	شرح متدهای کلاس BookDAL
۱۰۴۹.....FrmMember	شرح کلاس فرم	۱۰۱۷.....	کلاس AmanatDAL
۱۰۴۹.....FrmMember	شرح فیلدهای کلاس فرم	۱۰۱۷.....	شرح فیلدهای کلاس AmanatDAL
۱۰۴۹.....FrmMember	شرح توابع کلاس فرم		
۱۰۵۰.....FrmMember	شرح رویدادهای کلاس فرم		



شرح توابع کلاس فرم FrmNewAmanat ... ۱۱۰۱	فرم ویرایش اطلاعات اعضا ... ۱۰۵۸
شرح رویدادهای کلاس فرم FrmNewAmanat	طراحی فرم ویرایش اطلاعات اعضا ... ۱۰۵۸
..... ۱۱۰۱	شرح کلاس فرم FrmEditMember ... ۱۰۶۲
فرم ویرایش و حذف امانات ۱۱۰۶	شرح فیلدهای کلاس فرم FrmEditMember ... ۱۰۶۲
طراحی فرم ویرایش و حذف امانات ۱۱۰۶	شرح توابع کلاس فرم FrmEditMember ... ۱۰۶۳
شرح کلاس فرم FrmEditAmanat ۱۱۰۸	شرح رویدادهای کلاس فرم FrmEditMember
شرح فیلدهای کلاس فرم FrmEditAmanat ... ۱۱۰۸	 ۱۰۶۳
شرح رویدادهای کلاس فرم FrmEditAmanat	فرم ثبت کتاب ۱۰۷۴
..... ۱۱۰۹	طراحی فرم ثبت کتاب ۱۰۷۴
فرم ثبت کاربر ۱۱۱۹	شرح کلاس فرم FrmBook ۱۰۷۶
طراحی فرم ثبت کاربر ۱۱۱۹	شرح فیلدهای کلاس فرم FrmBook ۱۰۷۷
شرح کلاس فرم FrmNewUser ۱۱۲۲	شرح توابع کلاس فرم FrmBook ۱۰۷۷
شرح فیلدهای کلاس فرم FrmNewUser ... ۱۱۲۲	شرح رویدادهای کلاس فرم FrmBook ۱۰۷۷
شرح توابع کلاس فرم FrmNewUser ۱۱۲۲	فرم ویرایش و حذف کتاب ۱۰۸۱
شرح رویدادهای کلاس فرم FrmNewUser ۱۱۲۳	طراحی فرم ویرایش و حذف کتاب ۱۰۸۱
فرم ویرایش سطح دسترسی و نام کاربر ۱۱۲۷	شرح کلاس فرم FrmEditBook ۱۰۸۴
طراحی فرم ویرایش سطح دسترسی و نام کاربر	شرح فیلدهای کلاس فرم FrmEditBook ... ۱۰۸۴
..... ۱۱۲۷	شرح توابع کلاس فرم FrmEditBook ۱۰۸۵
شرح کلاس فرم FrmChangeAccess ۱۱۳۰	شرح رویدادهای کلاس فرم FrmEditBook ۱۰۸۵
شرح فیلدهای کلاس فرم FrmChangeAccess	فرم امانات ۱۰۹۲
..... ۱۱۳۰	طراحی فرم FrmListMember ۱۰۹۲
شرح توابع کلاس فرم FrmChangeAccess	شرح کلاس فرم FrmListMember ۱۰۹۳
..... ۱۱۳۰	شرح فیلدهای کلاس فرم FrmListMember ۱۰۹۳
شرح رویدادهای کلاس فرم FrmChangeAccess	شرح رویدادهای کلاس فرم FrmListMember
..... ۱۱۳۲	 ۱۰۹۴
فرم تغییر کلمه عبور کاربر ۱۱۳۶	طراحی فرم FrmListBook ۱۰۹۵
طراحی فرم تغییر کلمه عبور کاربر ۱۱۳۶	شرح کلاس فرم FrmListBook ۱۰۹۷
شرح کلاس فرم FrmChangePass ۱۱۳۷	شرح فیلدهای کلاس فرم FrmListBook ۱۰۹۷
شرح فیلدهای کلاس فرم FrmChangePass ۱۱۳۷	شرح رویدادهای کلاس فرم FrmListBook ۱۰۹۷
شرح رویدادهای کلاس فرم FrmChangePass	طراحی فرم امانات ۱۰۹۹
..... ۱۱۳۸	شرح کلاس فرم FrmNewAmanat ۱۱۰۰
فرم ارسال پیام ۱۱۴۰	شرح فیلدهای کلاس فرم FrmNewAmanat
طراحی فرم ارسال پیام ۱۱۴۰	 ۱۱۰۰

۱۱۶۲.....	افزافه کردن پارامتر	۱۱۴۱.....FrmNewMail	شرح کلاس فرم
۱۱۶۲.....	افزافه کردن فیلدها به گزارش	۱۱۴۲.....FrmNewMail	شرح فیلدهای کلاس فرم
۱۱۶۲.....	طراحی فرم نمایش کارت عضویت	۱۱۴۲.....FrmNewMail	شرح رویدادهای کلاس
۱۱۶۳.....	شرح کلاس فرم FrmRptKartMember	۱۱۴۶.....	فرم نمایش پیام
FrmRptKartMember	شرح فیلدهای کلاس فرم	۱۱۴۶.....	طراحی فرم نمایش پیام
۱۱۶۳.....		۱۱۴۷.....FrmShowMail	شرح کلاس فرم
شرح رویدادهای کلاس فرم		۱۱۴۷.....FrmShowMail	شرح فیلدهای کلاس فرم
۱۱۶۴.....	FrmRptKartMember	۱۱۴۸.....FrmShowMail	شرح توابع کلاس فرم
۱۱۶۵.....	فرم اصلی	۱۱۴۸.....FrmShowMail	شرح رویدادهای کلاس
۱۱۶۵.....	طراحی فرم اصلی	۱۱۵۲.....	گزارشات
۱۱۶۶.....	ساخت منو	۱۱۵۲.....	گزارش کتابهای هر گروه
۱۱۷۳.....	طراحی گزینههای SideBar1	۱۱۵۳.....	افزافه کردن فیلدها به گزارش
۱۱۸۰.....	شرح کلاس فرم FrmMain	۱۱۵۳.....	افزافه کردن پارامتر
۱۱۸۱.....	شرح فیلدهای کلاس فرم FrmMain	۱۱۵۳.....	طراحی فرم نمایش گزارش کتابهای هر گروه
۱۱۸۱.....	شرح رویدادهای کلاس فرم FrmMain	۱۱۵۴.....FrmRptBookByGroup	شرح کلاس فرم
۱۱۸۲.....	فرم ورود به برنامه	FrmRptBookByGroup	شرح فیلدهای کلاس فرم
۱۱۸۲.....	طراحی فرم ورود به برنامه	۱۱۵۵.....	
۱۱۸۳.....	شرح کلاس فرم FrmLogin	FrmRptBookByGroup	شرح رویدادهای کلاس فرم
۱۱۸۳.....	شرح فیلدهای کلاس فرم FrmLogin	۱۱۵۵.....	
۱۱۸۴.....	شرح توابع کلاس فرم FrmLogin	۱۱۵۶.....	گزارش لیست اعضا
۱۱۸۷.....	شرح رویدادهای کلاس فرم FrmLogin	۱۱۵۶.....	افزافه کردن فیلدها به گزارش
۱۱۹۰.....	شمای از Solution Explorer در پایان پروژه	۱۱۵۶.....	طراحی فرم نمایش گزارش لیست اعضا
۱۱۹۱.....	منابع	۱۱۵۷.....FrmRptMember	شرح کلاس فرم
		۱۱۵۷.....	گزارش کتابهای امانتی هر عضو
		۱۱۵۷.....	افزافه کردن پارامتر
		۱۱۵۷.....	افزافه کردن فیلدها به گزارش
		۱۱۵۸.....	طراحی فرم گزارش کتابهای امانتی هر عضو
		۱۱۵۹.....FrmRptAmanatMember	شرح کلاس فرم
		FrmRptAmanatMember	فیلدهای کلاس فرم
		۱۱۵۹.....	
			شرح رویدادهای کلاس فرم
		۱۱۶۰.....FrmRptAmanatMember	
		۱۱۶۲.....	کارت عضویت



سرآغاز

با ظهور دات نت و محیط توسعه گر VS و پس از آن به حاشیه رفتن تمامی محیط‌های تولید نرم‌افزار همچون دلفی محبوب، میکروسافت این بار با خود به رقابت می‌پردازد و در هر نسخه ای از دات نت ما شاهد پیدایش تکنولوژی‌ها و ایده‌های مختلف در جهت تسریع و تنظیم کیفیت تولید که منجر به نگهداری و توانایی توسعه‌پذیری آینده‌ی نرم‌افزارها می‌باشد، هستیم.

در نسخه Visual Studio 2008 این بار یکی از افراد نابغه که سابقاً در شرکت بورلند کار می‌کرد، به نام آقای آندرس هلسبرگ، زبان غیر روالمند LINQ را اختراع کرده و در زبان‌های برنامه‌نویسی دات نت همچون C#، Vb.Net قرار داد.

LINQ شاهکاری بود که امروزه آن را آینده دسترسی به داده‌ها می‌دانند. آقای هلسبرگ در یکی از مجلات مشهور جهانی، مرد نابغه سال ۲۰۰۹ نامیده شد. اما LINQ تنها پیشکشی دات نت ۲۰۰۸ نبود. در این نسخه تکنولوژی‌هایی همچون WCF، WF، WPF و Cardshape افزوده شد.

در Visual Studio 2010 ضمن تکمیل و توسعه‌ی تکنولوژی‌ها در جهت توانمندسازی آنها، امکانات محیط VS در مواردی چون خطایابی، ترسیم دیاگرام کلاس‌ها و UML نیز ارتقا پیدا کرده است. لازم به ذکر است که تغییراتی در زبان VB.Net نیز اعمال شده است که در طول کتاب با آنها آشنا خواهید شد.

در اینجا لازم می‌دانم شما را با بخش‌هایی از کتاب آشنا کنم:

بررسی مقدمات زبان برنامه‌نویسی VB.NET:

در این بخش ضمن مرور بر سنت زبان VB.NET، ساختارها و مفاهیم جدیدی که در VB2010 بوجود آمده نیز مورد بررسی قرار می‌گیرد.

جهش به دنیای برنامه نویسی شی گرا:

یک برنامه‌نویس کار خود را با آموختن الگوریتم و فلوچارت جهت توانائی در حل مسئله آغاز می‌کند و پس از خبرگی در حل مسائل، به سراغ یک زبان برنامه‌نویسی جهت کد کردن الگوریتم‌ها می‌رود. اما باید توجه داشته باشید که پس از طی این مراحل و رسیدن به مرحله‌ی توانائی در کد کردن، هنوز آموزه‌های پایه‌ای ملزوم برای برنامه‌نویس کامل نشده است، بلکه این بار یک برنامه‌نویس امروزی در جهت تولید نرم‌افزار، نیاز به یک جهش دارد و آن آشنائی و توانائی درک شی‌گرایی و برنامه نویسی شی‌گرا می‌باشد. زیرا شی‌گرا امروزه پایه و اساس تولید نرم‌افزار و تمامی تکنولوژی‌های دات نت است. از آنجا که همیشه تاکید بنده بر آن است که برنامه‌نویسان مفاهیم شی‌گرا را خوب یاد بگیرند تا بتوانند آنها را به درستی بکار بندند، فصلی کامل به زبانی ساده، همراه با مثال‌هایی از پیرامون زندگی واقعی نوشته‌ام و توصیه می‌شود به هیچ وجه تا این بخش را خوب درک نکرده و آن را یاد نگرفته‌اید، به سراغ



گام‌های بعدی نروید. زیرا مدامی که برنامه‌نویسی شی‌گرا را خوب بلد نباشید، نخواهید توانست با تکنولوژی‌های جدید همچون LINQ To SQL، Entity Framework، کار کنید. ضمناً در صورت عدم توانائی شما در برنامه‌نویسی شی‌گرا، در استفاده از خود محیط ویژوال استدیو نیز توانمند نخواهید شد. همچنین در صورت ناتوانی در این بخش، امکان پیاده‌سازی برنامه‌نویسی سه لایه که در این کتاب به سادگی بیان شده است نیز برایتان میسر نخواهد بود.

ابزارهای اضافه:

در هنگام اجرای پروژه‌ها، برنامه‌نویسان به امکاناتی بیش از امکانات فعلی موجود در محیط دات نت نیازمندند. لذا در گام کار با اکتویکس‌ها، موضوعاتی چون کار با فایل‌های PDF، SWF، استفاده از AnalogClock، MsAgent، تقویم شمسی و برنامه نویسی مدیریت ارسال و دریافت SMS، برنامه‌نویسی استفاده خودکار از نرم افزارهای Excel، Word و آشنائی با ابزارهای زیبا و تکمیل شده‌ی کنترل‌های قبلی دات نت به نام DevComponent آشنا خواهید شد.

زبان LINQ:

استفاده از زبان LINQ بسیار ساده است، زیرا این زبان نیازهای شما را بی‌آنکه الگوریتم آن را بدانید انجام می‌دهد. این زبان جهت استفاده در منابع داده‌ای مختلف مهیا شده است. لذا جهت جلوگیری از سخت شدن آموزش، کاربرد LINQ را پس از معرفی هر منبع داده‌ای، در فصلی جداگانه قرار داده ایم. در این فصل‌ها سعی شده زوایا و کاربرد این زبان در پروژه‌های بزرگ و واقعی مورد بررسی قرار بگیرد. در صورت مشکل در این زبان سعی کنید به فصل‌های مقدماتی آن مراجع نمائید. در این کتاب امکان انجام همه‌ی موضوعات هم در زمان طراحی و هم در زمان کدنویسی بیان شده است.

تولید نرم افزار اتوماسیون:

با توجه به نیاز روز افزون کشور در جهت تولید نرم‌افزارهای تجاری و رشد فوق العاده VS در این زمینه و با توجه به تغییر نگرش دسترسی به داده‌ها در NET. و کمبود منابع لازم جهت توجیه برنامه‌نویسان در استفاده از این معماری‌های جدید، لازم بر آن دانستم که گام کاملی مجتمع بر هشت فصل جهت بررسی بانک‌های اطلاعات و تکنولوژی‌های دسترسی به آنها را جهت آموزش در کتاب به گنجانم. در این بین با توجه به عدم وجود منابع فارسی در موضوعاتی همچون Entity Framework و برنامه نویسی سه لایه، سعی بر آن شده که مطالب به طور تئوری و عملی ارائه شوند.

پروژه:

مطابق عهده‌ی که با خود بسته‌ام، همانند کتاب قبلی در این کتاب نیز یک پروژه‌ی دیگر جهت اجرای مطالب آموزش داده شده تولید کردیم. به نظر می‌رسد این پروژه، راهنما و منبعی بسیار مناسب برای برنامه‌نویسان در زمینه‌ی برنامه‌نویسی شی‌گرا و سه لایه باشد.

در انتها از بزرگوارانی که مرا در این کتاب یاری داده‌اند به ترتیب زیر تشکر و قدردانی می‌کنم:
از سرکار گرامی خانم مهندس منصوره حاجی‌زاده که در تولید پروژه، فصل اکتویکس‌ها و ویراست علمی بسیار کمک نموده‌اند تشکر و کمال قدردانی را دارم.
از استاد بزرگوار جناب آقای مهندس حمزه پارس‌نژاد جهت طراحی جلد خلاقانه و زیبایشان ممنون و سپاسگذارم.

از آقای مهندس محمد رفیع نبی‌زاده جهت عکس‌برداری بسیار ممنونم.
در انتها امیدوارم توانسته باشم، گامی کوچک جهت تولید نرم‌افزارهای شی‌گرا و با استفاده از تکنولوژی‌های جدید برای برنامه‌نویسان عزیز ایرانی برداشته باشم.

بی‌صبرانه منتظر پیشنهادها و انتقادات شما خوانندگان عزیز هستم.

MohsenModhej@gmail.com

WWW.HowPrg.com

محسن مدحج

کتاب راهنمای کاربردی Visual Basic.Net 2010

سال ۱۳۸۹



فصل ۱



بررسی مقدمات زبان برنامه نویسی VB.NET





(Linus Torvalds)

صحبت کردن ساده است. کدت رو نشون بده.

مقدمات زبان برنامه نویسی VB.NET2010 :

هر زبان برنامه نویسی دارای قوانین و ساختار مربوط به خود در رابطه با تعریف متغیرها، شرطها، حلقهها و... می باشد که در ابتدای شروع به کار با آن زبان باید به سراغ آموختن این موضوعات برویم.

کلمات کلیدی :

کلمات ذخیره شده ای هستند که برای اهداف خاص مورد استفاده قرار می گیرند که به آنها کلمات کلیدی^۱ یا کلمات ذخیره شده می گویند، مانند کلمه کلیدی Dim که برای تعریف متغیرها به کار می رود.

در جدول زیر تعدادی از کلمات کلیدی را که در قسمت کلاسها، زیربرنامهها، شرطها، ساختارهای تکرار، نوع دادهها، استثناءها، رویدادها و آنهایی که به صورت عمومی بکار می روند را لیست کرده ایم:

Category	Keyword	Access	Modifier	Operator	Control	Language Extension
Label/Handler	Try	Dim	By Dim	Dim	End Try	Implements/NameSpace
Resource/Handler/Class	Finally	Const	Shared/Const	Dim	End Class	Class
Basic/Event	Finally	As	For/Next/Step/Each	Dim	By Val	Inherits/NotInheritable/Module/Interface
Conditional	When	Interface	While	While	By Ref	Interface Implementation
Handler	Do	Enum	Until	is	Call	New
Event/Event/Event	Then	Option	Do/Until	Nothing	Delegate	Property/Not Dim

شناسه: منظور از شناسه نامهایی است که به توابع، متغیرها و ... داده می شود (درحقیقت هویت یک چیز را بیان می کند). شناسهها به دو دسته ی شناسه های استاندارد و غیر استاندارد تقسیم می شوند:

الف) شناسه های استاندارد:

به منظورهای خاص توسط کامپایلر زبان VB.Net استفاده می شوند، مانند:

دیتا تایپها	توابع
Int16	Abs Round
Int32	Atan Pow
Int64	Cos Sqrt
UInt16	Sin Asc
UInt32	Tan InStr
UInt64	Exp Left

^۱ Reserved Word

ب) شناسه‌های غیر استاندارد:

کلمات معتبری هستند که نه جزء کلمات کلیدی و نه جزء شناسه‌های استاندارد هستند و برنامه نویس می‌تواند برای نام متغیرها، کنترل‌ها، توابع، برچسب‌ها و ... از آنها استفاده کند. با دانستن مفاهیم فوق، مقدمات زبان برنامه نویسی VB.NET را آغاز می‌کنیم.

متغیر چیست؟

محلی از حافظه‌ی اصلی (Ram) است که برای نگهداری انواع داده‌ها به کار می‌رود. همانطور که از نام متغیر مشخص است، امکان تغییر مقادیر درون آن در طول اجرای برنامه فراهم خواهد بود. برای تعریف متغیر دو مرحله‌ی اصلی در پیش رو داریم:

۱. نامگذاری متغیر (طبق قوانین)

۲. تعیین نوع متغیر

۱- نامگذاری متغیر طبق قوانین:

هر متغیر دارای نامی است که به وسیله‌ی آن می‌توان به متغیر مورد نظر دسترسی پیدا کرد. همانند انسان‌ها که برای دسترسی به آنها از نامشان استفاده می‌کنیم. اما این نامگذاری در هر زبان دارای قواعدی است که باید بر طبق آنها انجام شود. در ادامه، قوانین تعریف متغیر را بررسی می‌کنیم:

- اولین کاراکتر باید حرف یا زیر خط باشد. از کاراکتر اول به بعد می‌توانیم از اعداد و حروف نیز استفاده کنیم.

- نام متغیر نباید جزء کلمات کلیدی و شناسه‌های استاندارد باشد.

- در نام متغیر نباید از Blank (فاصله) و علائم خاص مانند ؟ ، ! ، \$ و ... استفاده کرد و حداکثر طول یک شناسه ۱۰۲۳ کاراکتر است.

علاوه بر قوانین نامگذاری ذکر شده بهتر است به هنگام نامگذاری متغیرها به نکات زیر نیز توجه کنید. با استفاده از این نکات می‌توانید خوانایی برنامه‌هایتان را افزایش دهید.

- در نام متغیر از پیشوندی استفاده کنید که نوع متغیر را مشخص کند. بدین ترتیب به وسیله‌ی نام متغیر از نوع آن هم مطلع خواهید شد و نیازی نیست به قسمت تعریف متغیر مراجعه کنید. در جدول زیر پیشوندهای برخی از نوع داده‌ها نشان داده شده است:

نوع داده	پیشوند	مثال
Boolean	Blн	Blн_flag
Byte	Byte	Byte_len
Short	Short	Short_number
Integer	Int	Int_code
Long	Lng	Lng_price
Single	Sng	Sng_avg
Double	Dbl	Dbl_scientific
String	Str	Str_firstname
Char	Chr	Chr_symbol

• از نام‌هایی استفاده کنید که متناسب با عملکرد متغیر باشند. به این ترتیب برنامه قابل فهم تر خواهد بود و نیاز کمتری به مستند سازی خواهد داشت. به طور مثال اگر متغیری را برای نگهداری نام افراد استفاده می‌کنید، می‌توانید از str_firstname برای نام آن استفاده نمایید.

برای جدا کردن قسمت‌های نام یک متغیر می‌توانید از حروف بزرگ و یا زیرخط استفاده کنید، مانند StrFirstName (به این روش نامگذاری شتری گفته می‌شود چون نام متغیر مانند کوهان شتر دارای پستی و بلندی می‌گردد.) و یا str_first_name. (به این ترتیب کاملاً مشخص است که یک متغیر رشته‌ای، برای نگهداری نام کوچک افراد تعریف نموده‌ایم.)



نامهای درست برای متغیر:

a2 , a2b , _2c

نامهای نادرست برای متغیر:

2a , a b , -2c , a?b

۲- تعیین نوع متغیر:

داده‌ها انواع مختلفی دارند. در این مرحله مشخص می‌کنیم که متغیر چه نوع داده‌ای (Data type) ^۲ را می‌تواند در خود جای دهد.

^۲ Data type : داده‌هایی که در کامپیوتر ذخیره می‌شوند، انواع مختلفی دارند. این نوع‌ها را Data type می‌نامیم.

انواع داده‌ها:

الف- داده‌های عددی

ب- داده‌های غیر عددی



الف- داده‌های عددی:

داده‌های عددی بر دو قسم می‌باشند:

۱. داده‌های عددی صحیح

۲. داده‌های عددی اعشاری

جدول نوع داده‌های عددی صحیح:

نوع داده (data type)	عمق داده حافظه اشغال شده (بر حسب بایت)	محدوده داده ای
Integer, Int32	4	-2'147'483'648 u 2'147'483'647
Short, Int16	2	-32'768 u 32'767
Long, Int64	8	-9'223'372'036'854'775'808 u 9'223'372'036'854'775'807
Byte	1	0 u 255
Sbyte	1	-128 u 127
UInteger, UInt32	4	0 u 4,294,967,295
ULong, UInt64	8	0 u 18,446,744,073,709,551,615 (1.8...E+19)
UShort, UInt16	2	0 u 65,535

جدول نوع داده‌های عددی اعشاری:

نوع داده (data type)	مقدار حافظه اشغال شده (بر حسب بایت)	محدوده داده‌ای
Float	4	برای مقادیر مثبت 1.401298E+45 یا 1.401298E+38 برای مقادیر منفی -1.401298E+38 یا -1.401298E+45
Double	8	اعداد بزرگ
Decimal	16	اعداد بسیار بزرگ

ب- داده‌های غیر عددی:

داده‌های غیر عددی عبارتند از:

- ۱- داده‌های کاراکتری
- ۲- داده‌های رشته‌ای
- ۳- داده‌های منطقی

جدول نوع داده‌های کاراکتری و رشته‌ای:

نوع داده (data type)	محدوده داده‌ای
Char	۱ کاراکتر
String	۲ میلیون کاراکتر

جدول نوع داده‌های منطقی و آبجکت:

نوع داده (data type)	مقدار حافظه اشغال شده (بر حسب بایت)	محدوده داده‌ای
Boolean	۱	T or F
Object	۴ یا ۸	هرنوع داده‌ای

اکنون با دانستن مفاهیم فوق می‌توان گفت: تعریف یک متغیر یعنی نامگذاری صحیح آن و تعیین نوع داده‌ای که می‌تواند ذخیره کند.

چگونگی تعریف متغیر:

- شکل کلی تعریف یک متغیر به صورت زیر است:

```

نوع متغیر as نام متغیر Dim
یا
نوع متغیر as ... , نام متغیر ۲ , نام متغیر ۱ Dim
و یا
... , نوع متغیر as نام متغیر ۲ , نوع متغیر as نام متغیر ۱ Dim
    
```

example ✓

```
Dim shr_len as short
```

در این مثال shr_len خانه‌ای از حافظه است که فقط می‌تواند اعداد ۳۲,۷۶۸- تا ۳۲,۷۶۷ را نگهداری کند.

- 32768 =< a < 32767

example ✓

```

Dim StrFirstName , StrLastName as String
Dim StrFirstName as String , ShrLen as Short
    
```

برخی از نوع داده‌های معروف و پرکاربرد، دارای کاراکتر شناسه نوع داده^۳ مخصوصی هستند.

به عنوان مثال :

Integer	%
Long	&
Single	!
Double	#
Decimal	@
String	\$

و شکل کلی استفاده از آنها در تعریف متغیر به صورت زیر است:

```

کاراکتر شناسه نوع داده + نام متغیر Dim
یا
... , کاراکتر شناسه نوع داده + نام متغیر , کاراکتر شناسه نوع داده + نام متغیر Dim
    
```

^۳ Data-Type Suffix



```
Dim StrLastName$ , SngAvg! , IntCode%
```



گاهی این روش بهترین راه حل برای تعریف چند متغیر از نوع‌های مختلف است.



همیشه باید بسته به نوع داده، Data type ای را در نظر بگیریم که کمترین میزان حافظه را اشغال نماید.



دو متغیر با شرایط مشخص شده تعریف نمائید:

```
0=<a<255 , b>255
```

پاسخ درست :

```
Dim a as byte
Dim b as short
```

پاسخ نادرست :

```
Dim a,b as short
```



وقتی نوع متغیر مشخص نشود به صورت خودکار از نوع داده‌ای Object در نظر گرفته خواهد شد و در این صورت امکان نگهداری هر نوع داده‌ای در آن فراهم می‌شود.

به شکل زیر:

نام متغیر Dim



```
Dim a
A=100
A="sss"
A=2.33
```

انتساب داده به متغیر:

عمل مقداردهی به متغیرها را دستور انتساب داده می‌نامیم. این عمل به دو شکل انجام می‌شود:
الف) در زمان تعریف (مقداردهی اولیه) به شکل زیر:

```
Dim مقدار یا متغیر = نوع متغیر as نام متغیر
```

example ✓

```
Dim a As Byte = 100
```

ب) در طول برنامه به شکل کلی زیر:

```
مقدار یا متغیر = نام متغیر
```

example ✓

```
Dim a As Byte  
a = 100
```



داده‌ها باید با نوع متغیرها و محدوده‌ی آنها سازگاری داشته باشند. در مثال زیر یک نمونه از مقداردهی نادرست متغیر را می‌بینیم:

```
Dim A as byte  
A = 1000
```



در مثال بالا a را در محدوده‌ی byte تعریف کرده‌ایم، پس نمی‌توانیم به آن عدد ۱۰۰۰ را نسبت دهیم.

اگر نکته شماره ۱ رعایت نشود دو نوع خطا به وجود می‌آید:

۱- اگر داده با نوع متغیر هماهنگ نباشد خطای زمان^۴ اجرا را مشاهده می‌کنیم.

example ✓

```
Dim A as short  
A = "salam"
```

^۴ خطای زمان اجرا، خطایی است که در هنگام اجرای برنامه رخ داده، باعث توقف برنامه می‌شود و سپس خطی که خطا در آن رخ داده است مشخص می‌گردد.

در این حالت خطای زمان اجرا را در هنگام مقداردهی به این متغیر مشاهده می‌کنیم. علت این خطا، آن است که در VB.Net ، متغیرها در زمان اجرا ساخته و مقداردهی می‌شوند.

۲- اگر داده در محدوده‌ی متغیر نباشد، دو حالت ممکن است پیش بیاید:

حالت اول : مستقیماً به یک متغیر مقداری خارج از محدوده بدهیم که در این صورت خطای کامپایلری^۵ را مشاهده می‌کنیم.



```
Dim A as byte
A = 4000
```

حالت دوم : متغیری را به آن انتساب دهیم که در محدوده متغیر اولیه نباشد، در اینصورت خطای زمان اجرا رخ می‌دهد.



```
Dim B as integer , A as byte
B=3276
A=B
```



داده‌های صحیح در متغیر اعشاری می‌تواند قرار بگیرد، اما عکس این قضیه یعنی قرار دادن داده‌های اعشاری در متغیر صحیح باعث می‌شود که قسمت اعشاری داده حذف گردد. به این نوع تبدیلات که خود کامپایلر VB.Net انجام می‌دهد، تبدیلات ضمنی می‌گویند.



```
Dim A As Single = 9.987
A = 99
Dim b as byte
B=9.987 / عدد ۹ می‌شود/
```



انواع متغیرهایی که در محدوده یکدیگرند و هم نوع هستند می‌توانند در یکدیگر قرار گیرند.

^۵ خطای کامپایلری، خطایی است که در زمان کدنویسی رخ می‌دهد و با کشیدن خط در زیر کد، محل خطا مشخص می‌شود .



```
Dim a, b as short
A=20
B=A
```



در vb.net اعدادی که نوشته می‌شوند به طور پیش فرض از نوع Integer می‌باشند.

● استفاده از نوع داده‌های کاراکتری و رشته‌ای:

● داده‌های کاراکتری^۶:

داده‌های کاراکتری فقط شامل یک کاراکتر هستند و می‌توانند شامل اعداد، حروف و علامتها باشند. اگر بیش از یک کاراکتر در یک متغیر کاراکتری قرار دهیم، بقیه‌ی کاراکترها حذف می‌شوند.



```
Dim chr_symbol as char
chr_symbol = "?"
chr_symbol = "b"
chr_symbol = "2"
chr_symbol = "Iran" → chr_symbol = "I"
```



در صورتی که قصد داشته باشید خود کاراکتر دابل کتیشن را به عنوان یک داده نگهداری نمائید باید دو بار این کاراکتر را تایپ کنید.



```
Dim chr_symbol as char
chr_symbol = ""
```

● داده‌های رشته‌ای^۷:

داده‌های رشته‌ای مجموعه‌ای از کاراکترهای به دنبال هم هستند. این کاراکترها می‌توانند حرف، علامت

^۶ Character
^۷ String

و حتی عدد باشند، مقادیر رشته‌ای و کاراکتری همیشه بین دو علامت دابل کوتیشن (") قرار می‌گیرند.



```
Dim str$
Str = "mohsen"
Str = "4537"
```



اگر تعداد کاراکترها بیش از دو میلیون، شود بقیه کاراکترها حذف می‌شوند.

ثابتها :

ثابت‌ها نیز مانند متغیرها برای نگهداری داده‌ها استفاده می‌شوند با این تفاوت که پس از تعریف مقدار ثابت، دیگر مقدار آن در طول برنامه تغییر نمی‌کند و ثابت در حقیقت نامی مجازی برای یک داده است که به طور مکرر در برنامه مورد استفاده قرار می‌گیرد و به شکل‌های کلی زیر قابل استفاده می‌باشد:

مقدار مورد نظر = نوع داده as نام متغیر Const
یا
Const مقدار مورد نظر = نام متغیر



در شکل اول باید همان اصول انتساب داده به متغیر رعایت شود، اما در شکل دوم می‌توان هر مقداری را نسبت داد.



```
Const a as byte = 19
Const a! = 20
Const a = 21
```

ثابت‌ها به شما امکان جایگزینی نام‌هایی را که به خاطر آوردنشان آسان است به جای مقادیر لیترالی که به خاطر آوردنشان دشوار است را می‌دهند. این کار می‌تواند نوشتن و نگهداری کد شما را تسهیل نماید.

● کلمه کلیدی Nothing :

هنگام تعریف یک متغیر، آن متغیر بسته به نوع داده‌ای که از آن تعریف شده است دارای یک مقدار پیش فرض خواهد بود. به عنوان مثال متغیر نوع Boolean دارای مقدار False و Integer دارای مقدار صفر (۰) می‌باشد.



پس از تعریف یک متغیر تا قبل از مقداردهی، آن متغیر دارای مقدار پیش فرض می‌باشد.

در هنگام کدنویسی در شرایطی که قصد ندارید مقدار خاصی را به یک متغیر (حال از هر نوعی) انتساب دهید کافی است، کلمه کلیدی Nothing را به آن انتساب دهید تا آن متغیر مقدار پیش فرض خود را ذخیره نماید.



```
Dim a As Integer
a=8
a = Nothing
Dim s As String
s="Ahmad"
s = Nothing
Dim b As Boolean
b = Nothing
Dim c As Font = Nothing
```

● نوع Nullable :

متغیرهایی از نوع Integer, Double و...، در صورتی که مقداری درون آنها وجود نداشته باشد، مقدار صفر را به صورت پیش فرض نگهداری می‌کنند. اما توجه داشته باشید که خود صفر یک عدد است و مقدار تهی نیست حال جهت آنکه این امکان پدید آید که اینگونه متغیرها مقدار تهی را نیز بتوانند نگهداری کنند از نوع Nullable به شکل کلی زیر استفاده می‌کنیم:

Dim نام متغیر As New Nullable(Of نوع داده)



```
Dim a As New Nullable(Of Integer)
```


نکته

در شکل فوق متغیر a در صورتی که هیچ مقداری به آن نسبت داده نشده باشد دیگر مقدار صفر را به طور پیش فرض نگهداری نمی‌کند بلکه دارای مقدار تهی یا هیچ خواهد شد. جهت دادن مقدار تهی به اینگونه متغیرها از کلمه کلیدی Nothing استفاده می‌شود.

مثال

```
a=Nothing
```



در صورتی که متغیری عددی به صورت معمول (غیر Nullable) تعریف شود و مقدار Nothing را به آن اختصاص دهیم مقدار تهی را نگهداری نمی‌کند. مثلاً در نوع Integer بازهم مقدار صفر نگهداری خواهد شد.

روشی دیگر جهت تعریف یک متغیر از نوع Nullable وجود دارد، برای این منظور به راحتی می‌توانید با اضافه کردن یک علامت سوال در پایان نوع داده اینکار را انجام دهید.

مثال

```
Dim a As Integer?  
Dim b As Double?  
Dim c As Byte?
```

داده‌های شمارشی :

جهت درک راحت‌تر اینگونه داده‌ها تصور کنید، تعدادی از ثابت‌های عددی مرتبط با یکدیگر را در یک گروه قرار داده‌اید.

به شکل کلی زیر:

```
Enum نام As  
مقدار عددی = عضو ۱  
مقدار عددی = عضو ۲  
End Enum
```

حال جهت استفاده از آنها کافی است که نام گروه و سپس نام ثابت مورد نظر را صدا بزنید:

به شکل کلی زیر:

```
عضو مورد نظر. نام داده شمارشی
```

```
Enum روزهای_هفته As Integer
```

```
    شنبه = 0
```

```
    یکشنبه = 1
```

```
    دوشنبه = 2
```

```
    سه_شنبه = 3
```

```
End Enum
```

همانطور که مشاهده می‌کنید در مواردی که مجبور به قرار دادن فاصله بودیم از علامت زیرخط (_) استفاده کردیم.



در این مثال جهت اشاره به این مطلب که امکان تعریف شناسه‌های فارسی نیز وجود دارد از حروف فارسی استفاده کردیم لذا پیشنهاد نمی‌شود در کد برنامه‌ها از حروف فارسی استفاده شود.



داده‌های شمارشی باید خارج از زیربرنامه (Sub, Function) تعریف شوند.



یک ساختار داده‌ی شمارشی را می‌توان به عنوان یک نوع داده استفاده نمود و از آن متغیر تعریف کرد.



```
Enum روزهای_هفته As Integer
```

```
    0 = شنبه
```

```
    1 = یکشنبه
```

```
    2 = دوشنبه
```

```
    3 = سه_شنبه
```

```
End Enum
```

```
Sub Main()
```

```
    Dim a As روزهای_هفته
```

```
    a = دوشنبه, روزهای_هفته
```

```
End Sub
```

نوع داده ساختاری :

این نوع داده‌ها توسط برنامه نویسان تعریف می‌شوند و توسط آن می‌توان یک مجموعه از متغیرها با نوع‌های مختلف را در یک گروه با یک عنوان ذخیره نمود به عنوان مثال می‌توان یک ساختار به نام Personel ایجاد کرده و متغیرهای با عناوین نام، نام خانوادگی و سن را در آن قرار داد. جهت تعریف یک ساختار از شکل کلی زیر استفاده می‌شود:

```
نام ساختار Structure
    نوع متغیر As نام متغیر Dim
    نوع متغیر As نام متغیر Dim
    ...
End Structure
```



```
Structure Personel
    Dim Name As String
    Dim Fmily As String
    Dim age As integer
End Structure
```



از Structure همانند داده‌های شمارشی نمی‌توان به طور مستقیم استفاده نمود بلکه باید آن را به عنوان یک نوع داده تصور کرده و متغیری از این نوع داده جدید تعریف نمائیم:

```
Dim prsl As Personel
prsl.age = 1
prsl.Name = "fardin"
prsl.Fmily = "Salami"
```

همانطور که مشاهده می‌کنید ابتدا یک متغیر از این نوع داده‌ی جدید تعریف کرده و جهت استفاده از اعضای درون آن، پس از نام متغیر یک نقطه گذاشته و سپس عضو مورد نظر را انتخاب می‌کنیم.

● کلمه کلیدی With :

همانطور که در مثال فوق مشاهده می‌کنید برای دسترسی به عضوهای ساختار پرسنل هر بار باید ابتدا نام آن را تایپ کرده تا بتوانیم به عضو مورد نظر دسترسی پیدا کنیم جهت سهولت کار و جلوگیری از تایپ مجدد نام ساختار از عمگر With به شکل کلی زیر استفاده می‌کنیم:

```
With نام متغیر ساختار
    مقدار مورد نظر = عضو.
    مقدار مورد نظر = عضو.
End With
```

example

```
Dim prsl As Personel
With prsl
    .age = 1
    .Name = "fardin"
    .Fmily = "Salmi"
End With
```

● آشنایی با محیط Console Application :

حال وقت آن است که آموخته‌های خود را در زبان VB.Net عملی کنیم اما به علت آنکه هنوز محیط ویژوال را نشناخته‌ایم، نمی‌توانیم در محیطی که حاوی اشیاست برنامه نویسی کنیم. بنابراین از Console که در VS.Net وجود دارد و جهت ایجاد برنامه‌های شبیه به سیستم عامل DOS^۸ با کدنویسی VB.Net استفاده می‌نماییم.

^۸ - منظور از DOS همان Command Prompt ویندوز است که برنامه‌های تحت داس را شبیه سازی می‌کند. اینگونه برنامه‌ها فقط دارای خروجی متنی هستند و هیچگونه عنصری در آنها وجود ندارد.

نخستین باری که VS را اجرا می کنید پنجره ای نمایش داده می شود (تصویر ۱-۱). مطابق با شکل زبان Visual Basic Development Settings را انتخاب کنید. اینکار سبب می شود یک سری پیش فرضها در محیط VS برای زبان مورد نظرتان در نظر گرفته شود.



تصویر ۱-۱

اکنون مراحل زیر را برای ایجاد برنامه Console دنبال کنید.
برای این کار به ترتیب File → New → Project را برگزینید (تصویر ۱-۲).



تصویر ۱-۲

سپس از کادر نمایش داده شده گزینه Console Application را انتخاب کنید.



با فشردن کلید ok شکل زیر را مشاهده می کنید.

```
Module Module1
    Sub Main ()

    End Sub
End Module
```

شروع کدنویسی در Console :

توجه داشته باشید که تمامی کدها را باید بین Sub Main و End Sub بنویسید.
در ابتدا باید کد زیر را وارد کنید:

```
Console.ReadLine
```

علت نوشتن این کد آن است که پنجره Console بعد از اجرا بلافاصله بسته می شود و در واقع شما با نوشتن این کد به پنجره دستور می دهید که تا کاربر Enter نکرده است بسته نشود.

دستورهای ورودی / خروجی :

دستور ورودی:

دستور Console.ReadLine برای گرفتن ورودی از کاربر استفاده می شود. در شکل زیر توسط این دستور متغیر a مقداردهی شده است:

```
a = console.readline ()
```

example 1

```
Module Module1
  Sub Main ()
    a = console.readline ()
  End Sub
End Module
```

در این مثال a مقداری می‌شود و بعد پنجره بسته می‌شود. برای اینکه این شرایط پیش نیاید باید بلافاصله کد مربوطه (یعنی Console.readline()) را تایپ کنید. به کد زیر دقت کنید:

```
Module Module1
  Sub Main ()
    a = console.readline ()
    Console.readline()
  End Sub
End Module
```

برای اجرای برنامه دکمه‌ی (Start Debugging ) یا کلید F5 را کلیک کنید. بدین ترتیب شکل زیر را مشاهده خواهید کرد:



example 2

مقداری را که وارد می‌کنید باید هم نوع و هم محدوده با نوع متغیر باشد.

example 3

```
Dim a as integer=console.readline()
```



برای وارد کردن مقادیر رشته‌ای و کاراکتری، همانند دستور انتساب نیاز نیست از علامت دابل کوتیشن("") استفاده شود.

دستور خروجی:

برای چاپ کردن مقادیر و متغیرها بر روی صفحه‌ی نمایش از شکل کلی زیر استفاده می‌کنیم:

```
Console.WriteLine (مقدار یا متغیر مورد نظر)
```



```
Console.WriteLine ("Hello My Friend")
```

عملگرها :

یک عبارت شامل مجموعه‌ای از مقادیر، متغیرها و ... می‌باشد. برای ارتباط هر یک از این اجزا از عملگرها استفاده می‌کنیم و آن وقت همین متغیرها عملوند به حساب می‌آیند (به عبارت دیگر به طرفین هر عملگر، عملوند گفته می‌شود). عملگرهای مختلفی وجود دارند که کارهای مختلفی انجام می‌دهند مانند عملگر (+) که برای جمع دو عدد در مقادیر عددی استفاده می‌شود.

عملگرها به چند دسته تقسیم می‌شوند :

۱- عملگرهای محاسباتی : جمع (+)، منها (-)، ضرب (*)، توان (^)، تقسیم با اعشار (/)، تقسیم صحیح (\)، mod (باقی مانده)

۲- عملگرهای منطقی و بیتی : And , Or , Xor , Not , AndAlso , OrElse

۳- عملگرهای رابطه‌ای یا مقایسه‌ای : =, <, >, <=, >=, <>

۴- عملگرهای ترکیبی : &=, |=, /=, *=, +=

۵- عملگرهای رشته‌ای : &+, +=

عملگر تقسیم صحیح (\) : مقدار یا متغیر سمت چپ را به مقدار یا متغیر سمت راست تقسیم می‌کند که جواب همواره عددیست صحیح (از قسمت اعشاری صرف نظر می‌شود). اما در صورت استفاده از عملوندهای اعشاری، آنها گرد می‌شوند.

example ✓

dim A as byte

A=12 \ 10 → A=1

عملگر تقسیم اعشاری (/): مقدار یا متغیر سمت چپ را به مقدار یا متغیر سمت راست تقسیم می‌کند و جواب همواره عددیست اعشاری.

example ✓

dim A as single

A=12/10 → A=1.2

عملگر mod: عدد سمت چپ را بر عدد سمت راست تقسیم می‌کند و باقی مانده را به عنوان جواب بازمی‌گرداند. جواب آن همواره عددیست صحیح.

example ✓

Dim A as byte

A=12 mod 10 → A=2

• مثالی از عملگرهای ترکیبی:

Dim A as byte

A=5

A+=5 → A=10

A -= 2 → A=8

A *= 2

A \= 2

A /= 2

A ^= 2



عملوندها باید با یکدیگر و نیز با عملگرها سازگاری داشته باشند. همچنین باید توجه داشته باشید که مقدار بازگشتی از عبارات همواره با نوع و محدوده متغیر در برگیرنده‌اش سازگاری داشته باشد.



نوع هر عبارت هم‌نوع با عملوندهای همان عبارت تعیین خواهد شد.

example ✓

اگر نوع داده عملوندها در یک عبارت یکسان باشد:

Byte + Byte → Byte

Integer * Integer → Integer

تفاوتی نمی‌کند که عملگر در عبارات چه چیزی باشد.

اگر نوع داده عملوندها در عبارت متفاوت باشد آنگاه نوع داده با محدوده بیشتر به عنوان نوع داده کل عبارت در نظر گرفته می‌شود:

Integer + byte → Integer

Integer * byte → Integer



نتیجه‌ی عبارات عددی باید در محدوده‌ی عملوندهایشان (اعضای عبارت) باشد.

example ✓

```
Dim b, b1 As Byte, i As Integer
```

```
b = 100
```

```
b1 = 200
```

```
i = b + b1
```

Error



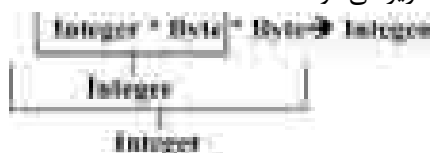
به صورت ساده می‌توان گفت که در خط آخر، خطا در سمت راست عبارت اتفاق افتاده است یعنی قبل از آنکه نتیجه در متغیر I قرار گیرد خطا رخ می‌دهد، علت آن است که عبارت دارای عملندهای b و b1 از نوع Byte است و با توجه به نکته فوق نوع داده‌ای کل عبارت همان Byte در نظر گرفته می‌شود. حال آنکه خروجی عبارت برابر با ۳۰۰ بوده و از محدوده‌ی ۲۵۵ خارج است.

حال خطای بالا را به شکل زیر برطرف می‌کنیم:

```
i = 1 * b * b1
```



در این جا عملوند ۱ در حاصل عبارت بی‌تاثیر بوده اما با توجه به آنکه در VB.Net اعداد به تنهایی از نوع Integer در نظر گرفته می‌شود این روش باعث رفع خطا خواهد شد. نوع داده عبارت قبل به صورت زیر می‌شود:



عملگرهای مقایسه‌ای :

نتیجه عباراتی با عملگرهای مقایسه‌ای جوابیست همواره منطقی (True یا False) و معمولا در دستورات شرطی استفاده می‌شوند. در جدول زیر عملگرهای مقایسه‌ای لیست شده اند:

عملگر	کاربرد
=	مساوی
<	کوچکتر از
<=	کوچکتر یا مساوی
<>	نامساوی
>	بزرگتر
>=	بزرگتر یا مساوی



2>33 → False
2<34 → True
2<>34 → true
2=34 → false



می‌توان خروجی عبارتی با این عملگرها را در متغیر Boolean ذخیره کرد.

Dim b as Boolean = 2<=2
B=2<>5

عملگرهای منطقی :

نتیجه‌ی عبارتی با عملگرهای منطقی جوابیست همواره منطقی.

عملگر NOT: مقدار مخالف ارزش عملوند را بر می‌گرداند. در صورتی که شرط مورد نظر ارزش درست را داشته باشد، NOT آن مقدار نادرست را بر می‌گرداند و برعکس.

نتیجه‌ی عبارات با عملوند NOT:

X	Not (X)
T	F
F	T

عملگرهای AND یا ANDALSO:

عملگر AND در صورت درست بودن تمامی عملوندها مقدار TRUE را می‌دهد. نتیجه‌ی عملگر ANDALSO نیز همانند عملگر AND است با این تفاوت که اگر اولین عملوند ارزش False داشته باشد مقدار False را می‌دهد و دیگر دومین عملوند را بررسی نمی‌کند. بنابراین سرعت آن بیشتر از عملگر AND می‌باشد.

نتیجه‌ی عبارات با عملگر AND یا ANDALSO:

X	Y	X AND Y	X ANDALSO
F	F	F	F
F	T	F	F
T	F	F	F
T	T	T	T

عملگرهای OR و ORELSE:

عملگر OR در صورت درست بودن یکی از عملوندها، مقدار TRUE را می‌دهد. نتیجه‌ی عملگر ORELSE همانند عملگر OR است با این تفاوت که اگر اولین عملوند ارزش True داشته باشد مقدار True را می‌دهد و دیگر دومین عملوند را بررسی نمی‌کند و سرعت آن بیشتر از عملگر OR می‌باشد.

X	Y	X OR Y	X ORELSE Y
F	F	F	F
F	T	T	T
T	F	T	T
T	T	T	T

نتیجه‌ی عباراتی با عملگر XOR:

در صورتی که تعداد عملوندهای صحیح، فرد باشد خروجی آن TRUE و در غیر اینصورت FALSE می‌باشد:

X	Y	X XOR Y
F	F	F
F	T	T
T	F	T
T	T	F

اولویت عملگرها :

عملگرها نسبت به یکدیگر دارای تقدم و اولویت هستند. در عباراتی که شامل چندین عملگر می‌باشند، این مسئله حائز اهمیت است. اگر در یک عبارت چند عملگر هم اولویت باشند عملیات از چپ به راست صورت می‌گیرد.

ترتیب اولویت عملگرها به شکل زیر است:

- ۱- توابع و پرانتزها
- ۲- توان
- ۳- علامت مثبت (+) و منفی (-) و not
- ۴- ضرب (*) و تقسیم (/)
- ۵- تقسیم صحیح (÷)
- ۶- Mod (باقی مانده تقسیم)
- ۷- جمع و منها
- ۸- عملگرهای مقایسه‌ای شامل: <, <=, >, >=

دستورهای شرطی :

هر برنامه‌ای ممکن است سه نوع ساختار کنترلی داشته باشد تا مراحل اجرای آن را کنترل نمایند. این سه نوع ساختار عبارتند از: ترتیب - تصمیم - تکرار

ساختار دستور IF :

هنگامی که نیاز به تصمیم‌گیری باشد از دستورات شرطی استفاده می‌شود. در این ساختار e یک عبارت یا متغیر شرطی است.

If e Then		}
دستور یا دستورات	1	}
Else		}
دستور یا دستورات	2	}
End If		

وجود قسمت ۲ کاملاً اختیاری و بر حسب نیاز می‌باشد.

program

برنامه‌ای بنویسید که دو عدد بگیرد و اگر مجموع آنها کمتر از ۱۰ بود "One" را چاپ کند و در غیر این صورت "More" را چاپ نماید.

<pre> Module Module1 Sub Main() Dim a As Byte Dim b As Byte Console.WriteLine("Please type First Number:") a = Console.ReadLine Console.WriteLine("Please type Next Number:") b = Console.ReadLine If a + b < 10 Then Console.WriteLine("One") Else Console.WriteLine("More") End If Console.ReadLine() End Sub End Module </pre>	تعریف متغیرها چاپ پیام مقدار دهی متغیر IF ساختار دستور برای اینکه تا Enter نزده‌اید پنجره بسته نشود.
--	---

program

برنامه‌ای بنویسید که اگر اعداد بزرگتر از ۱۰ در متغیر قرار داده شود، پیام long و در غیر این صورت پیام "small" را نمایش دهد (با استفاده از یک متغیر منطقی).

```

Sub Main()
  Dim A As Byte
  Dim b As Boolean
  A = Console.ReadLine
  b = A > 10
  If b=True Then
    Console.WriteLine("long")
  Else
    Console.WriteLine("small")
  End If
  Console.ReadLine()
End Sub
End Module

```

● ساختار If – Else نردبانی :

یک ساختار معروف به شرح زیر است:

```
If e1 then
S1
Else
  If e2 then
    S2
  Else
    If e3 then
      S3
    End If
  End If
End If
```

● ساختار If-Elseif :

با استفاده از این روش ساختار قبلی را به آسانی می‌توان پیاده‌سازی کرد. همچنین استفاده از این ساختار خوانایی برنامه را نیز افزایش می‌دهد.

```
If e1 then
S1
ElseIf e2 then
S2
ElseIf e3 then
S3
End if
```

برنامه‌ای بنویسید که یک عدد را از ورودی گرفته و اگر برابر ۱۰ بود چاپ کند "Ten"، اگر برابر ۲۰ بود چاپ کند "Twenty" و اگر برابر ۳۰ بود چاپ کند "Thirty" و از برنامه خارج شود.

```
Module Module1
Sub Main()
    Dim a As Byte
    a = Console.ReadLine
    If a = 10 Then
        Console.WriteLine("Ten ")
    Else
        If a = 20 Then
            Console.WriteLine("Twenty ")
        Else
            If a = ۳۰ Then
                Console.WriteLine("Thirty ")
            End
        End If
    End If
    Console.ReadLine()
End Sub
End Module
```

دستور خروج از برنامه

دستور End سبب خروج از برنامه می‌شود.

دستور Case :

زمانی که چندین شرط داشته باشیم یا به عبارتی بیش از سه دستور If بخواهیم به کار ببریم منطقی‌تر آن است که از دستور Case به شکل کلی زیر استفاده نماییم:

```
Select Case e
Case مقدار مورد نظر
    دستور یا دستورات
Case مقدار مورد نظر
    دستور یا دستورات
Case Else
    دستور یا دستورات
End Select
```



```
Dim i As Integer
i = Console.ReadLine
Select Case i
    Case 1
        Console.Write("Rayane")
    Case 2
        Console.Write("Markazi")
    Case 3
        Console.Write("VB.NET")
    Case Else
        Console.Write("Linq")
End Select
```



در صورتی که هیچ کدام از مقادیر ۱ یا ۲ یا ۳ نباشد مقدار "Linq" چاپ می‌شود. در صورت تمایل می‌توانید از الگوهای زیر نیز استفاده کنید.

بررسی اعداد در محدوده‌ای معین:

محدوده بالا To محدوده پایین Case



Case 1 To 22

جهت تعیین مقدار در وضعیت مورد نظر:

Case is شرط



در صورتی که مخالف عدد ۲۲ باشد.

در صورتی که برابر با عدد ۲۲ باشد. Case Is > 22

● حلقه‌های تکرار :

به کمک حلقه‌ها می‌توان تعدادی دستورالعمل را به دفعات معین و یا تا پدید آمدن شرایطی خاص تکرار کرد. حلقه‌ها را می‌توان به کمک دستور `if` و دستور پرش `goto` ساخت. اما تاکید زبان‌های ساخت یافته بر عدم استفاده از این نوع حلقه‌هاست.

● حلقه For (با تعداد تکرار مشخص) :

برای تکرار تعدادی دستور به دفعات معین از حلقه For استفاده می‌شود. فرم کلی حلقه For به دو شکل زیر است:

الف) تعریف متغیر شمارنده قبل از حلقه:

نوع متغیر Dim Counter as [مقدار Step] مقدار نهایی To مقدار اولیه = For Counter دستور یا دستورات حلقه Next Counter	
--	---

ب) تعریف متغیر شمارنده در ساختار حلقه:

[مقدار Step] مقدار نهایی To مقدار اولیه = نوع متغیر For Counter as دستور یا دستورات حلقه Next Counter	
---	---

Counter : شمارنده حلقه می‌باشد.

بدیهی است که در شکل ب پس از پایان کار حلقه متغیر شمارنده از بین خواهد رفت.

- در صورتی که مقدار اولیه در حلقه کوچکتر از مقدار نهایی باشد، به این حلقه، حلقه‌ی افزایشی گفته می‌شود.



```
For i=1 to 10
  Console.WriteLine(i)
Next i
```

در این مثال اعداد از ۱ تا ۱۰ به ترتیب نمایش داده می‌شوند.

- در صورتی که مقدار اولیه در حلقه بزرگتر از مقدار نهایی باشد، به این حلقه، حلقه‌ی کاهش‌ی گفته می‌شود.



```
For i=10 to 1 step -1
  Console.WriteLine(i)
Next i
```

در این مثال اعداد از ۱۰ تا ۱ نشان داده می‌شوند. در حلقه‌های کاهش‌ی مقدار Step عددی منفی می‌باشد.



در حلقه افزایشی اگر مقدار نهایی از مقدار اولیه کوچکتر باشد حلقه اصلاً اجرا نمی‌شود. به همین ترتیب در حلقه for کاهشی نیز اگر مقدار نهایی بزرگتر از مقدار اولیه باشد حلقه اجرا نمی‌شود. در این مورد خطایی هم صورت نمی‌گیرد و فقط کنترل اجرای برنامه به خط بعد از حلقه می‌رود.



اگر مقدار اولیه شمارنده حلقه For با مقدار نهایی آن برابر باشد (چه حلقه افزایشی باشد چه کاهشی) دستورات درون این حلقه فقط یک بار اجرا می‌شود.



در VB.NET هنگامی که حلقه تمام می‌شود شمارنده حلقه یکی بیشتر از مقدار نهایی حلقه است.



مقدار نهایی حلقه فقط یکبار آن‌هم در ابتدای حلقه مشخص می‌شود اما مقدار اولیه حلقه در گردش‌های بعدی تغییر خواهد کرد، بنابراین مقدار نهایی حلقه در بدنه حلقه قابل تغییر نخواهد بود اما مقدار اولیه حلقه در بدنه قابل تغییر است.



در صورت آنکه قصد داشته باشید تعداد تکرار یک حلقه در زمان اجرای برنامه تعیین شود به دو شکل این عمل صحیح خواهد بود:

```
Dim k as integer
K=console.readline()
For I as integer=0 to k
```

دستورات

Next

```
For I as integer=0 to console.readline()
```

دستورات

Next



همانطور که در نکته‌ی قبل گفتیم، مقدار نهایی حلقه فقط یکبار تعیین شده و بنابراین دستور ورودی فقط یکبار اجرا می‌شود.



برنامه ای بنویسید که اعداد ده تا بیست را چاپ کند.

```
Module Module1
```

```
Sub Main()
```

```
Dim a As Byte
```

```
For a = 10 To 20
```

```
Console.WriteLine(a)
```

```
Next
```

```
Console.ReadLine()
```

```
End Sub
```

```
End Module
```



شمارنده حلقه یکی یکی افزایش می یابد و ده بار عدد a را نمایش می دهد.

تعریف متغیر شمارنده در ساختار حلقه:

For i As نوع متغیر = مقدار اولیه To مقدار نهایی [مقدار Step]

دستور یا دستورات حلقه

Next



```
For i As Byte = 0 To 10 Step 2
```

```
Console.WriteLine("hello")
```

```
Next
```



پس از پایان حلقه دیگر امکان استفاده مجدد از این متغیر شمارنده وجود ندارد.

حلقه While :

در نوع دیگری از حلقه‌ها (به نام حلقه‌های شرطی) تعداد دفعات تکرار معلوم نیست و حلقه تا پدید آمدن شرط خاصی تکرار می‌شود، هر چند که گفته شد می‌توان این حلقه‌ها را با if و پرش goto ساخت، اما راحت‌تر این است که از حلقه While به شکل کلی زیر استفاده کنیم:

While e

دستور یا دستورات حلقه

End While



e : در این ساختار یک عبارت یا متغیر شرطی است.

program

برنامه‌ای بنویسید که پیام hello را ۱۰ بار چاپ کند (از حلقه while استفاده کنید).

```
Module Module1
    Sub Main()
        Dim i As Byte
        While i <> 10 }
            Console.WriteLine("hello")
            i += 1 }
        End While
        Console.ReadLine()
    End Sub
End Module
```

ساختار حلقه While

تا زمانی که I مخالف ده می باشد

شمارنده حلقه

program

برنامه ای بنویسید که تا عدد ۱۷ وارد نشده است برنامه بسته نشود.

```
Module Module1
    Sub Main()
        Dim a As Byte
        While a <> 17
            a = Console.ReadLine
        End While
    End Sub
End Module
```

مثال ۱۷-۱

در ابتدای حلقه a مخالف ۱۷ است و به همین دلیل تا زمانی که عدد ۱۷ وارد نشده است، دستور درون حلقه (a=Console.ReadLine) تکرار می‌شود. در نتیجه نیازی به استفاده از دستور Console.ReadLine نیست و a مرتباً مقادیر ورودی را در خود جای می‌دهد.

آرایه چیست؟

تعدادی متغیر از یک نوع و تحت یک نام می‌باشند. هر یک از متغیرهای درون آرایه با یک شماره که به آن اندیس^۹ می‌گوییم شناسایی می‌شود. متغیرهای درون آرایه را عناصر آرایه می‌نامیم. شکل کلی تعریف آرایه به صورت زیر است:

نوع متغیر As (تعداد اندیس) نام آرایه Dim

مثال ۱۷-۲

Dim a (4) as Byte



همانطور که می‌دانید شماره اندیس آرایه از صفر شروع می‌شود، پس دقت داشته باشید که تعداد عناصر آرایه یکی بیشتر از اندیسی است که شما داده‌اید. برای دسترسی به خانه‌های آرایه از الگوی زیر پیروی کنید:

(شماره اندیس) نام آرایه

مثال ۱۷-۳

a (0)
a (1)

برای مقداردهی به هر خانه‌ی آرایه از الگوی زیر پیروی کنید:

"مقدار مورد نظر" = (شماره اندیس) نام آرایه

^۹ Index.

example ✓

a (0)=1
a (2)=20

۱		۲۰		
---	--	----	--	--



در اینجا نیز باید اصول مقداردهی رعایت شود. چرا که هر خانه در واقع یک متغیر

است.

program ✓

برنامه‌ای بنویسید که اعداد ۱ تا ده را در یک آرایه قرار داده و نمایش دهد.

```
Dim a(9) As Byte
Dim i As Byte
For i = 0 To 9
    a(i) = i + 1
    Console.Write(a(i))
Next
```

آرایه بدون پهنای باند :

برای تعریف آرایه بدون پهنای باند به شکل زیر عمل می‌کنیم:

Dim نام آرایه () As نوع متغیر

example ✓

Dim a() As Byte

جهت مقداردهی به این نوع آرایه‌ها از شکل زیر استفاده می‌شود:

Dim نام آرایه () As نوع متغیر = { مقدار اول و مقدار دوم و ... }

example ✓

Dim A () As Byte = { 10, 20, 9, 6, 4 }

حال با توجه به آنکه پنج مقدار به آرایه‌ی بدون پهنای باند داده‌ایم، این آرایه حاوی پنج عنصر و دارای اندیسهای ۰ تا ۴ می‌باشد.

در VB10 آرایه‌های بدون پهنای باند را به شکل زیر نیز می‌توان تعریف نمود:

Dim نام آرایه { مقدار ۱, مقدار ۲, مقدار ۳, ... }



```
Dim a = {1, 2, 3}
```

جهت اطلاع از طول آرایه های بدون پهنای باند، می توان از متد Count استفاده نمود.



```
Console.WriteLine(a.Count)
```



در زبانهای VB9 و ما قبل از آن در صورتی که نوع متغیر یا آرایه همانند بالا مشخص نشود، به صورت خودکار، نوع آن Object در نظر گرفته می شود و این در حالی است که نوع آرایه ی بالا در vb10 از نوع Integer خواهد بود.
در موارد دیگر به مثالهای زیر توجه کنید:

```
Dim b = {1, 2, 3.5} 'از نوع Double تعریف می شود'
```

```
Dim c = {"1", "2", "3"} 'از نوع String تعریف می شود'
```

```
Dim d = {1, "A"} 'از نوع Object تعریف می شود'
```

حال با استفاده از یک حلقه ی For به عناصر این آرایه دسترسی پیدا می کنیم:

```
For i As Byte = 0 To 4
    Console.WriteLine(A(i))
Next
```

● حلقه For Each :

این حلقه جهت گردش در یک مجموعه به کار می رود، به این طریق که در هر بار گردش حلقه، به ترتیب تک تک اعضا در شمارنده حلقه قرار می گیرد.
به شکلی کلی زیر:

```
For Each   مجموعه in شمارنده
NEXT
```



```
Dim A() As Integer = {10, 20, 9, 6, 4}
```

```
For Each d In A
    Console.WriteLine(d)
Next
```


مجموعه متغیرهای شمارنده

در گردش اول مقدار ۱۰ در شمارنده d و در گردش دوم مقدار ۲۰ در شمارنده d قرار می‌گیرد و در ادامه نیز اینکار مرتب انجام می‌شود تا آنکه مقدار ۴ در متغیر d قرار گرفته و کار حلقه به پایان می‌رسد.



در اکثر مواقع نیاز نیست متغیر شمارنده از قبل تعریف شده باشد، زیرا شمارنده به صورت خودکار هم نوع با مجموعه تعریف می‌شود. اما در صورت نیاز متغیر شمارنده را به دو شکل زیر می‌توان تعریف کرد:

الف) در ساختار حلقه:

```
مجموعه In نوع متغیر شمارنده As نام متغیر شمارنده
Next
```

example

```
Dim s() As String = {"من", "ی", "ا", "د", "خ"}
```

```
For Each j As String In s
    Console.WriteLine(j)
Next
```

مجموعه متغیرهای شمارنده

تک تک اعضای مجموعه‌ی S که از نوع رشته هستند نمایش داده می‌شود.

ب) قبل از تعریف بدنه حلقه:

```
Dim s() As String = {"کی", "تا", "کجا", "چور", "چه"}
```

```
Dim j As String
For Each j In s
    Console.WriteLine(j)
Next
```

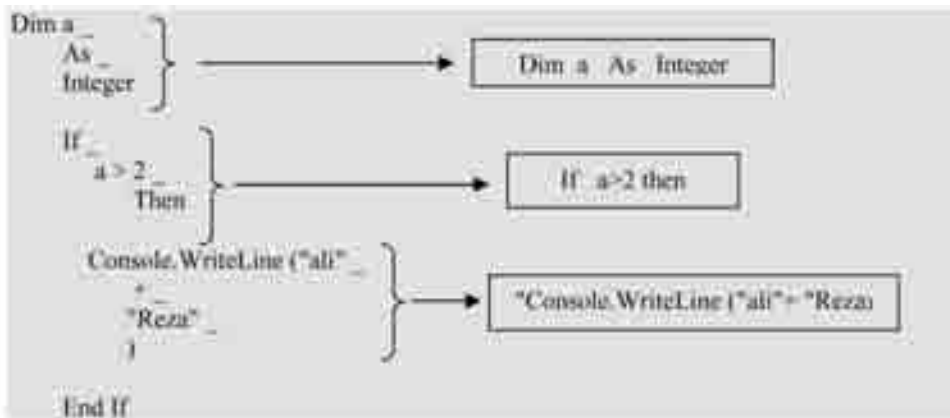
● شکستن دستورات :

در صورتی که دستورات طولانی شوند و قصد داشته باشید یک سری دستورات بهم پیوسته را در چند خط بشکنید، می‌توانید از علامت _ استفاده نمائید، برای درک این موضوع به مثال‌های زیر توجه نمائید.

ابتدا کدهای زیر را در نظر بگیرید:

```
Dim a As Integer
If a > 2 Then
    Console.WriteLine("ali" + "Reza")
End If
```

حال کدهای بالا را به چندین خط تقسیم می‌کنیم:



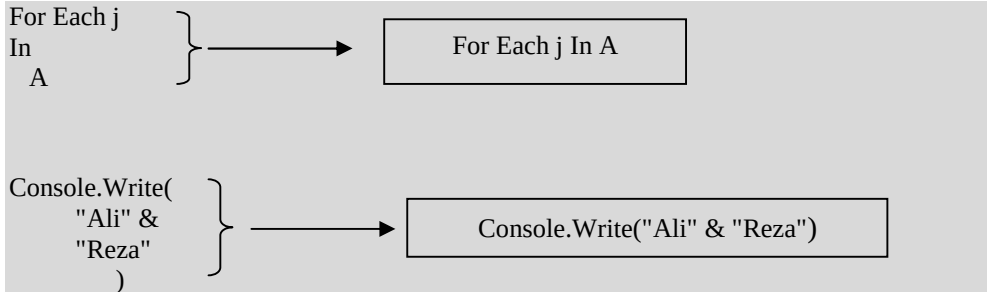
ادامه خطوط به صورت ضمنی:

در VB10 (Visual Studio 2010) این امکان پدید آمده است که در برخی از قطعه دستورات بی آنکه علامت _ را استفاده نمود بتوان کدها را در چندین خط شکست. علائم و کلمات کلیدی که قبل یا بعد از آن می توان خطوط را بی آنکه از علامت زیر خط (_) استفاده نمود شکست، به شرح زیر است:

علائم	قبل	بعد	علائم	قبل	بعد
,		X	/ Operator		X
(		X	\ Operator		X
)	X		Mod Operator		X
{		X	+ Operator (unary and binary)		X
}	X		^= Operator		X
<%=		X	*= Operator		X
%>	X		/= Operator		X
<		X	\= Operator		X
>	X		+= Operator		X
Aggregate	X	X	-=Operator		X
Distinct	X	X	&= Operator		X
From (in a query)	X	X	< Operator		X
From (in a collection initializer)		X	<= Operator		X
Group By	X	X	> Operator		X
Group Join	X	X	>= Operator		X
Join	X	X	= Operator		X
Let	X	X	<> Operator		X
Order by	X	X	Is Operator		X
Select (in query context)	X	X	IsNot Operator		X
Skip	X	X	Like Operator		X
Skip While	X	X	& Operator		X
Take	X	X	And Operator		X
Take While	X	X	Or Operator		X
Where	X	X	Xor Operator		X
In	X	X	AndAlso Operator		X
Into	X	X	OrElse Operator		X
On	X	X			
Ascending	X	X			
Descending	X	X			
^ Operator		X			
* Operator		X			



به مثال‌های زیر توجه نمائید:



توضیحات (comment) :

برنامه نویسان معمولاً در قسمت‌های مختلف کدهایشان توضیحاتی را جهت یادآوری عملکرد آن قطعه کد و یا جهت خوانایی بیشتر برای سایر برنامه نویسان به برنامه اضافه می‌کنند. این توضیحات گاهی می‌تواند یک قطعه کد ساده نیز باشد. ابتدا یک تک کتیشن (') تایپ کرده و سپس توضیحات مورد نظر را تایپ می‌نمائیم. به شکل زیر:

توضیحات'



'dar in ghesmat process marbot be search anjam mishavad
'in ghesmat hamanand code For I as byte=0 to 5 amal khahad kard



توضیحات توسط مترجم کامپایل نمی‌شود.