مقدمه:

*VB*از زبانهای برنامه نویسی تحت *Windows* می باشد که برای کدنویسی از دستورات زبان *Basic* سود می برد. *(VB*مانند تمام زبانهای برنامه نویسی تحت ویندوز*)* با استفاده از تمام امکانات زیبای ویندوز (که باعث فراگیرشدن این سیستم عامل زیبا و توانمند در میان کاربران شده است)، طراحی محیطی زیبا و قدرتمند را برای پروژه مورد نظر، بسیار ساده می نماید.
هنرجویان عزیز توجه داشته باشند که برای فراگیری*VB*، آشنایی با ویندوز و *Basic* لازم است. همچنین بهتر است در هنگام مطالعه، *VB*باز باشد و مطالب را در همان لحظه کار کنند.

خاصیت کنترل ها و رویدادها:

اول کار با کنترل های *Visual Basic* را شروع می کنیم.

در *VB* دو نوع کنترل وجود دارد. کنترل هایی که جزو استاندارد *VB* است و کنترل های *ActiveX* که پسوند *OCX* دارد و به *Toolbar* افزوده می شود. خود *VB* تعدادی از این ها را برای برنامه فراهم کرده است. این فایل ها را می توان از اینترنت دانلود کرد. برای اضافه کردن آن ها به جعبه ابزار کلید راست فشار دهید و *Components* را برگزینید و با تیک زدن بر روی هر گزینه ای از ان به جعبه ابزار شما افزوده می شود.

خاصیت های مشترک بین کنترل:

گفتیم که هر کنترل خواصی برای خود دارد. یک سری از این خاصیت ها بین کنترل ها مشترک است. که تعدادی از آنها در زیر آمده است.
*Alignment*نحوه تراز کردن متن را تعیین می کند .
*BackColor*برای تعیین رنگ زمینه کنترل .
*BorderStyle*تعیین می کند که آیا کنترل حاشیه داشته باشد یا خیر.
*Caption*متنی که روی کنترل دیده می شود.
*Enable*تعیین کننده ی فعال یا غیر فعال بودن کنترل.
*Font*فونت متن داخل کنترل را تعیین می کند.
*ForeColor*رنگ متن داخل کنترل را تعیین می کند.
*Height*ارتفاع کنترل
*Left*فاصله لبه ی چپ کنترل از فرم را مشخص می کند.
*MousePointer*شکل کرسر موس وقتی که بر روی کنترل قرار می گیرد را مشخص می کند.
*Name*نامی که در برنامه کاربرد دارد.
*ToolTipText*متنی که وقتی موس را چند ثانیه روی کنترل متوقف کنیم ظاهر می شود.
*Top*فاصله لبه ی بالای کنترل از فرم را مشخص می کند.
*Visible*تعیین کننده مرئی یا نامرئی بودن کنترل .
*Width*عرض کنترل

روال رویداد:

در ویندوزرویداد های *(Events)*زیادی رخ می دهد. ویندوز این رویداد ها را بررسی می کند که این رویداد مربوط به کدام برنامه ی در حال اجراست. در این موقع به برنامه خبر می دهد که این رویداد اتفاق افتاده است. اکثرا این رویدادها مربوط به موس یا کیبورد است. مثل کلیک وتایپ متن. ما می توانیم با نوشتن تابعی به این رویداد ها پاسخ مناسبی بدهیم. به این معنی که ما تابعی می نویسیم که اگر رویداد خاصی رویداد برنامه دستور های ما را اجرا کند. مثلا با کلیک روی دکمه فرمان از برنامه خارج شود.
هر کنترل رویداد های خاصی برای خود دارد. البته بعضی از آنها نیز مشترک است. هر کنترل می تواند تابعی برای هر رویداد خود داشته باشد. مثلا یک دکمه فرمان می تواند روالی برای رویداد کلیک و روالی برای رویداد کلیک راست بنویسیم. برای هرکنترل نیز باید یک تابع نوشت. مثلا اگر برنامه سه دکمه فرمان داشته باشد برای هر یک باید یک روال نوشت.

رویدادهای مهم:

Active قتی روی می دهد که فرم فکوس را در اختیار گیرد.
Click وقتی روی می دهد که بر روی شیئی یک بار یکی از کلید های موس را بزنیم .اینکه کدام کلید زده خواهد شد قابل تشخیص است.
DblClick وقتی روی می دهد که روی شیئ دو بار کلیک شود.
Deactive وقتی روی می دهد که فرم فکوس را از دست می دهد .
Initialize وقتی روی می دهد که فرم برای بار اول به وجود می آید.
Load وقتی روی می دهد که فرم وارد حافظه فعال می شود و فرم ظاهر می شود.
Paint وقتی روی می دهد که ویندوزمجبور می شود قسمتی از فرم را مجددا ترسشیم کند.
Resize وقتی روی می دهد که کاربر اندازه فرم را تغییر دهد.
Unload وقتی روی می دهد که فرم از حافظه فعال خارج شود.
Change وقتی روی می دهد که کاربر متن جعبه متن را تغییر دهد.

متغیرها

گاهی لازم است در طول برنامه تان عددی یا داده ای را در جایی ذخیره کنید و روی آن عملیاتی انجام دهید .برای این کار به متغییر نیاز دارید. متغییرها همان طور که از نامشان پیداست خانه هایی از حافظه اند که می توانند مقداری را در خود نگه دارند که این مقدار قابل تغییر است. محتوای این خانه ها تا وقتی که مقدار جدیدی را به آنها نسبت نداده ایم در طول برنامه ثابت می ماند. اگر مقدار متغییر تغییر کند مقدار قبلی پاک شده و مقدار جدید در آن قرار می گیرد *vb*.بر خلاف اکثر زبانهای برنامه نویسی مثل سی و پاسکال کاربر را مجبور به تعریف متغییر نمی کند. این ویژگی *vb* ممکن است برای اکثر برنامه نویسان تازه کار به معنی رهایی از دردسرهای تعریف متغییر باشد ولی در عمل ممکن است مشکلات زیادی را بوجود آورد (اگر در طول برنامه نام متغییر را اشتباه تایپ کنید کامپایلر پیام خطایی به شما نمی دهد و آن را به عنوان متغییرجدیدی به حساب می آورد و این یعنی فاجعه پیشنهاد میشود قبل از انجام هر کاری در *vb* کلمه *Option Explicit* را در قسمت بالای برنامه تان بنویسید

Option Explicit .به ویژوال بیسیک می گوید کاربر خود تمام متغییرهای مورد نیازش را تعریف می کند، بنابراین اگر از این به بعد متغییری را به اشتباه تایپ کنید پیام خطایی دریافت می کنید. برای اینکه *vb* به طور خودکار *Option Explicit*را به بالای برنامه هایتان اضافه کند گزینه *Require Variable Declaration* را از منوی *Tools/Options*انتخاب کنید.

مجموعه و علائم و نشانه های مجاز : مانند : اعداد ۰ تا ۹ - حروف کوچک یا بزرگ *A* تا *Z* و * / () و ...

- **متغیر** : مکانی با نام معین از حافظه برای نگهداری داده ها که در طول برنامه قابل تغییرند.
- **ثابتها** : مکانی با نام معین از حافظه برای نگهداری داده ها که در طول برنامه قابل تغییر نیستند .

نکته مهم : ثابتها به دو دسته بدون نوع و نوع دار تقسیم می شوند . **تفاوت** این دو در این است که مقدار ثوابت نوع دار در طول برنامه قابل تغییر است.

مقدار ثابت = نوع ثابت *as* نام ثابت *Const* یا مقدار = نام ثابت *Const*

Const N as integer =100 *Const P=3.14*

- **شناسه** : اسمی است که برنامه نویس به دلخواه خود برای نام گذاری متغیرها ، ثابتها ، عنوان برنامه ها و ... انتخاب می کند .

نکته ۱ : حداکثر طول یک شناسه در *VB* **۲۵۵** کاراکتر می باشد .

نکته ۲ :شناسه می تواند فقط شامل حروف الفبایی و ارقام ۰ تا ۹ و علامت زیر خط (_) باشد . بکاربردن علائم ویژه مجاز نیست

نکته ۳ : اولین حرف نباید از نوع عددی باشد .

نکته ۴ :استفاده از فاصله غیر مجاز است .

نکته ۵: *VB* در استفاده نام شناسه از حروف کوچک یا بزرگ تفاوتی قائل نمی شود .

عملگرها و انواع آنها :

عملگر : نماد هایی هستند برای انجام عملیات خاص – عملگرها برای انجام اعمال خاص روی عملوندها بکار می روند.

انواع عملگرها	
محاسباتی	+ - * / \ *Mod*
رابطه ای	> >= < <= = <>
منطقی	*Eqv Imp And Or Xor Not*
بیتی	*Shr Shl And Or Xor Not*
رشته ای	+ &

تقدم عملگرها		
1	پرانتز	*()*
2	توان	^
3	تفریق یکانی	-
4	ضرب و تقسیم	* /
5	تقسیم صحیح	\
6	باقیمانده	*Mod*
7	جمع و تفریق	+ -

انواع عملگرها:

عملگرهای شرطی =، < ، > ، <>، <. > . < > پاسخ این عملگر ها یا *True* است یا *False* .

بوسیله کلمات زیر می توانید عبارت شرطی زیر را با هم ترکیب کنید.

And هر دو شرط باید درست باشد تا پاسخ *True* شود . *Or* اگر یکی از شرط درست باشد پاسخ *True* می شود.

Xor فقط یک شرط باید درست باشد نه دو شرط تا پاسخ *True* شود.

*Not True*را تبدیل به *False* و *False* را به *True* تبدیل می کند.

ترتیب اجرای این کلمات در موقع اجرا به این ترتیب است:

۱. *Not*
۲. *And*
۳. *Or*
۴. *Xor*
۵. *Eqv*
۶. *Imp*

نکته مهم : اولویت عملگرها

۱. **ریاضی**
۲. **مقایسه ای**
۳. **منطقی**

انواع داده ها در *VB* :

انواع داده های عددی در *VB*			
میزان حافظه مصرفی		محدوده	نوع
۸ بیت	۱بایت	*0* تا *255*	*Byte*
۱۶ بیت	۲بایت	*-32768* تا *32767*	*Integer*
۳۲ بیت	۴بایت	*-2147483648* تا *2147483648*	*Long*
32 بیت	*4*بایت	*0* تا *65535*	*Single*
۶۴ بیت	۸بایت		*Double*
64 بیت	۸بایت		*Currency*

پسوندهای عددی در *VB*	
Integer	*%*
Long	*&*
Single	*!*
Double	*#*
Currency	*@*
String	*$*

انواع داده های غیر عددی در *VB*		
میزان حافظه مصرفی	محدوده	نوع
طوا رشته	۱ تا ۶۵۴۰۰ نویسه	*String* (با طول ثابت)
طول رشته + ۱۰ بایت	۰ تا ۲ میلیارد نویسه	*String* (با طول متغیر)
۸ بایت	از اول ژانویه ۱۰۰ تا ۳۱ دسامبر ۹۹۹۹	*Date*
۲ بایت	*True* یا *False*	*Boolean*
۴ بایت	معادل شی تهریف شده	*Object*
۱۶ بایت	هر عددی تا*Double*	*Variant* عددی
طول رشته + ۲۲	مانند*String*	*Variant* متنی

خواص *Label* ها:

خاصیت های *Label* در پنجره ی *properties* وجود دارد.

Auto size این خاصیت در صورتی که *true* باشه باعث میشه تا اندازه *label* به طور اتوماتیک به اندازه متن اون باشه و اگر *false* باشه اندازه به صورت دستی قابل تغییر است.

Back Style اگر این خاصیت مساوی ۰ یا *transparent* باشه *label* شفاف میشه و فقط متن اون پیدا میشود و اگر ۱ یا *Opaque* باشه *label*به صورت مات و غیر شفاف در میاد که رنگ اون توسط خاصیت *Back color*قابل تغییر است

BorderStyle گر مقدار این خاصیت ۰ یا *None* باشه *label* به صورت تخت *Flat* تبدیل میشه و اگر مقدارش مساوی *1* یا *Fixed Single*باشه *label* به صورت ۳بعدی نمایش داده میشه.

ToolTipText این هم توضیحی است که با رفتن موس روی *label* نمایش داده میشه.

خواص *text box*:

خاصیت های *textbox* در پنجره ی *properties* وجود دارد.

Name این رو که حتما می دونید که اسم اون هستش ولی نکته ای که باید به اون توجه کرد اینه که برنامه نویس های حرفه ای در اول اسم *textbox*ها *txt* روهم مینویسند مثلا *txtdisplay* یا . *txtname*

Alignment این خاصیت نحوه تراز شدن متن رو در *textbox* رو نشون می ده (وسط چین،چپ چین،راست چین)

Appearance این خاصیت به شکل و شمایل اون مربوط می شه اگر *flat* رو انتخاب کردید *textbox* به صورت تخت در میاد ولی اگر *3D* رو انتخاب کنید به صورت سه بعدی و تو رفته در میاد .

Backcolor ای هم مربوط میشه به رنگ داخل .*textbox*

enable هم مربوط میشه به فعال یا غیر فعال بودن .

Forecolor این هم رنگ متن داخلش رو مشخص می کنه .

locked این خاصیت مشخص می کند که آیا کاربر می تواند متن داخل ان را عوض کند یا نه .

Maxlenght این خاصیت حداکثر تعداد کاراکترهایی را که می توان درون *textbox* وارد نمود مشخص می کند .

Multiline چنانچه این خاصیت *true* باشد وقتی در *textbox* در حال تایپ کردن هستید اگر اینتر را فشار دهید به خط بعد می روید و می توانید متن را در بیشتر از یک خط بنویسید ولی اگر این خاصیت *false* باشد فقط میتوانید در یک خط متن را وارد کنید .

Passwordchar حتما دیدید که وقتی در حال تایپ کردن پسورد هستید هرچی تایپ می کنید به جاش یه علامت ستاره تایپ میشه که به خاطر اینه که متن به صورت محرمانه باشه وکسی نتونه ببینه.حالا شما در مقابل این خاصیت هر کاراکتری رو وارد کردی به جای متن اون کاراکتر نشون داده میشه.مثلا اگر ۳ رو وارد کردید، هرچی تایپ کردید بجای اون ۳ میبندید .

RightToLef این خاصیت مشخص می کنه متن از چپ به راست نوشته شود یا از راست به چپ .

scrollbar چنانچه بخواهید میله های مرور روی جعبه متن ظاهر شوند ونیز برای تعیین تعداد انها باید از این خاصیت استفاده کنید.مقدار ۰ *None*-از ظاهر شدن میله های مرور جلوگیری میکند.مقدار ۱ *Horizontal*-فقط یک میله مرور افقی نشان می دهد.مقدار ۲- *vertical*اجازه نمایش میله مرور عمودی را می دهد..مقدار ۳ *both*-هر دو میله مرور افقی و عمودی را نشان می دهد .

Text این خاصیت متن اولیه(مقدار پیش فرض) را که درون جعبه متن ظاهر می شود را نشان می دهد.

خواص فرم ها:

appearance این خاصیت مشخص میکند که فرم به صورت سه بعدی (۳ *(D*باشد یا تخت *.(flat)*

Back Color این خاصیت رنگ زمینه فرم را مشخص میکند.

Border style این خاصیت اگر بر روی(۰*(None*-باشد فرم را بدون حاشیه و دکمه های مینیمایز و ماکسیمایز وبستن نشان میدهد و کاربر نمی تواند آن را تغییر اندازه بدهد و اگر بر روی(۱*(Fixed single*-باشد فرم را با حاشیه و دکمه بستن نشان میدهد و کاربر نمی تواند آن را تغییر اندازه بدهد و اگر بر روی(۲ *(Sizable*-باشد تمام دکمه ها و حاشیه فرم را نشان میدهد.

Icon این خاصیت آیکون برنامه را مشخص می کند.

Max button این خاصیت فعال یا غیر فعال بودن دکمه ماکسیمایز را مشخص می کند.

Min button این خاصیت فعال یا غیر فعال بودن دکمه مینیمایز را مشخص می کند

Mouse icon این خاصیت شکل نشانگر موس را تعیین می کند

Mouse Pointer این خاصیت نوع شکل نشانگر موس را مشخص می کند مثل ساعت شنی یا دست شدن نشانگر

Movable این خاصیت مشخص میکند که آیا کاربر اجازه دارد که فرم را جابجا کند یا نه

Picture عکس زمینه فرم را مشخص می کند

ShowIn Taskbar مشخص می کند که برنامه در تسکبار دیده شود یا نه

Startup position محل قرار گرفتن فرم در هنگام شروع برنامه را مشخص می کند

Window state نوع نمایش پنجره در هنگام شروع برنامه(مینیمایز/ماکسیمایز/نرمال(

کنترل دکمه فرمان: (*Command Button*)

این کنترل می تواند از برنامه تقاضای کاری را بکند یا به آن فرمان دهد.

Caption عنوانی که بر روی دکمه دیده خواهد شد.

Picture می توان با این خاصیت تصویری را رو دکمه قرار داد.

Style وضعیت دکمه را نشان می دهد. اگر گزینه *Graphical* انتخاب نشود خاصیت *Picture* بی اثر است.

آشنایی با تابع:

در *VB* دو نوع تابع وجود دارد. یکی سابروتین که هیچ مقدار بازگشتی ندارد و دیگری فانکشن که یک مقدار بازگشتی. خود *VB* هم یکسری تابع داخلی دارد. مثلا تابع *LoadPicture( )* برای قرار دادن تصویر در ابزارهایی است که توانایی نمایش تصویر را دارند. ما هم می توانیم برای خود، تابع تعریف کنیم.

فایده تابع ها این است که می توان کد برنامه را دسته بندی کرد و هر وظیفه را به تابعی واگذار کرد.اشکال زدایی از برنامه آسان می شود. چون اگر برنامه در جایی خواسته ی ما را برآورده نکرد می دانیم این وظیفه مربوط به کدام قسمت است. اگر برنامه به کرات نیاز به اجرای یکسری کد داشته باشد می توان آن را در یک تابع نوشت و بعد آن را در مواقع مورد نیاز فراخوانی کرد. مثلا می توانیم تابعی برای رسم مستطیل بنویسیم که ابعاد آن را به عنوان ورودی دریافت کرده وآن را رسم کند.

*VB*فایل های خاصی را برای نوشتن توابع اختصاص داده است که به آن **ماژول** می گویند. با وجود این فایل ها می توانید تابع هایی را که کاربرد زیادی دارد در این فایل ها ذخیره کرده و در برنامه های مختلف از آنها استفاده کرد و در وقت صرفه جویی کرد. برای ایجاد این فایل ها روی پنجره *Project* کلید راست را فشار دهید و از قسمت *Add* ، *Module*را انتخاب کنید.

آشنایی با برخی از توابع داخلی:

LoadPicture این تابع یک تصویر را به برنامه اضافه می کند. در داخل پرانتز مسیر تصویر قرار می گیرد.مثل :

Image1.Picture = LoadPicture(“c:\Pic1.bmp”)

MsgBox تابعی است که یک جعبه پیام را به کاربر نشان می دهد.

همانطور که می بینید جعبه پیام یک آیکون، یک پیام و چند دکمه فرمان دارد. شکل کلی این دستور به صورت مقابل است.

RetVal = Msgbox (message [, type [,title]])
intResponse= MsgBox(strPrompt,intStyle,strTilte)

آرگومان اختیاری *Title*عنوان جعبه پیام را مشخص می کند اگر از این آرگومان استفاده نشود نام پروژه در عنوان جعبه پیام قرار می گیرد.

به مثال زیر توجه کنید:

IntVal = MsgBox("This Is an Example Message")

با اجرای این دستور جعبه پیامی به شکل زیر دیده می شود:

اگر آرگمان های دیگر را مشخص نکنیم تنها پیام با دکمه *OK* نشان داده می شود. آرگمان دوم نوع دکمه ها و آیکون ها را مشخص می کند . *strTitle*تیتر جعبه پیام را مشخص می کند . *intResponse* کد دکمه ای را که کاربر فشار داده در خود نگه می دارد که با دستور *If* یا *Select case*می توان آن را بررسی کرد .دکمه های این جعبه در لیست پایین آمده است:

vbOkOnly دکمه *OK*
vbOKCancel دکمه های *OK* و *Cancel*
vbAbortRetryIgnore دکمه های *Abort,Retry,Ignore*
vbYesNoCancel دکمه های *Yes,No,Cancel*
vbYesNo دکمه های *Yes,No*
vbRetryCancel دکمه های *Retry,Cancel*

آیکون هایی که می توان استفاده کرد به شرح زیر است:

vbCritical	یا کد ۱۶
vbQuestion	یا کد ۳۲
vbExclamation	یا کد ۴۸
vbInformation	یا کد ۶۴

بین دکمه و ایکون باید علامت + بگذاریم. مقدارهایی که جعبه پیام می تواند بر گرداند به شرح زیر است:

نام ثابت مقدار
vbOK 1
vbCancel 2
vbAbort 3
vbRetry 4
vbIgnore 5
vbYes 6
vbNo 7

مثال:

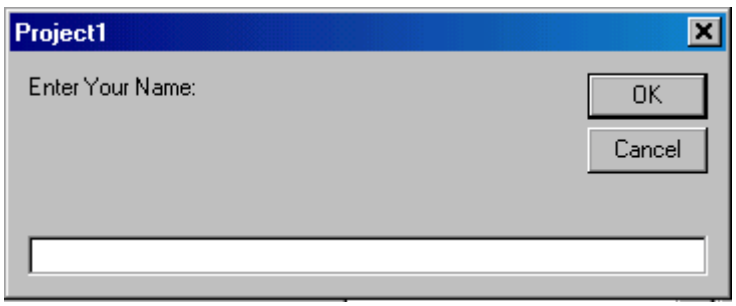
```
Dim I As Integer
I  = MsgBox("Do you want to exit?", vbYesNo + vbQuestion, "Exit")
If  I=6 Then End
```

InputBox جعبه ای که سوالی می کند و کاربر باید مقداری را وارد کند (یک دستور ورودی است)

شکل کلی تابع به صورت زیر است:

StrAns = InputBox (Message [, Title][, Default][, Xpos][,Ypos])
strAnswer = InputBox(مقدار پیش فرض, تیتر, پیام *)*

StrAns = InputBox ("Enter Your Name:")



توابع عددی:

Int(Value) محاسبه جزء صحیح
Fix(Value) اعشار را از عدد حذف می کند.
Log(Value) محاسبه لگاریتم طبیعی
Atn(Value) محاسبه آرک تانژانت
Cos(Value) محاسبه کسینوس
Sin(Value) محاسبه سینوس
Tan(Value) محاسبه تانژانت
Abs(Value) محاسبه قدر مطلق
Sqr(Value) محاسبه جذر
این توابع با عدد های داخل پرانتز سروکار دارند.

توابع تشخیص نوع :

IsEmpty() اگر متغییر داخل پرانتز مقدار نگرفته باشد *True* برمی گر داند.
IsNull() اگر مقدار متغییر داخل پرانتز *Null* باشد *True* برمی گر داند.
IsNumeric() اگر متغییر داخل پرانتز از نوع عددی باشد یا توانایی تبدیل به عدد را داشته باشد *True* برمی گر داند.

توابع تبدیل نوع:
این توابع برای تغییر نوع یک متغییر به نوع دیگر به کار می رود
مسئله مهمی که باید توجه کنید این است که باید **حدود انواع متغییر را در نظر بگیرید تا ازاین توابع انتظار درست کار کردن داشته باشید.**

CBool() تبدیل آرگمان به *Boolean*
CByte() تبدیل آرگمان به *Byte*
CCur() تبدیل آرگمان به *Currency*
CDbl() تبدیل آرگمان به *Double*
CDec() تبدیل آرگمان به *Decimal*
CInt() تبدیل آرگمان به *Integer*
CLng() تبدیل آرگمان به *Long*
CSng() تبدیل آرگمان به *Single*
CStr() تبدیل آرگمان به *String*
CVar() تبدیل آرگمان به *Variant*

توابع کار با رشته:

*Len()*تعداد حرف یک متغییر *String* را برمی گرداند.
*Right()*قسمتی از یک رشته را از سمت راست بر می گرداند.
*Left()*قسمتی از یک رشته را از سمت چپ بر می گرداند.
*Mid()*قسمتی از یک رشته را از وسط بر می گرداند
*UCase()*تمام حروف متن داخلش را تبدیل به حروف بزرگ می کند.
*LCase()*تمام حروف متن داخلش را تبدیل به حروف کوچک می کند .

توابع اسکی:
هر کاراکتر دارای یک کد است.توابع *Chr()* و *Asc()* برای تبدیل اعداد به کاراکتر و بلعکس هستند.

مثال:
A=Chr(65)
65=Asc("A")

استفاده از روال ها در برنامه نویسی:
بهترین روش برای نوشتن برنامه های بزرگ و پیچیده ، تقسیم آن به قسمتهای کوچکتر است ، بطوری که هر قسمت وظیفهٔ خاصی را انجام دهد. با انجام این عمل به خوبی می توان روی قسمتهای مختلف برنامه مدیریت کرده ، در صورت بروز خطا علت آن را راحت و سریع شناسایی و اصلاح نمود .

برای اعمال چنین روشی از روالها *(Routins)* استفاده خواهیم کرد .استفاده از روال ویژگیهای زیر را دارد :
کیفیت مدیریت برنامه ها را افزایش می دهد.
کارآیی نرم افزار را بالا می برد و می توان روالهای ایجاد شده را در برنامه های دیگر هم بکار برد و یا از روالهای ایجاد شدهٔ قبلی ، در برنامه ها استفاده کرد .
از تکرار در کدنویسی برنامه ها جلوگیری می کند و براحتی در هر جایی از برنامه که نیاز باشد ، می توان با استفاده از نام روال آن را فراخوانی کرد
آزمایش و خطایابی و تغییر روالها به مراتب آسانتر است
استفاده از روالها باعث افزایش خوانایی برنامه می شود
در انتقال اطلاعات به داخل روال از آرگومانها استفاده می شود . آرگومانها متغیرهایی هستند که در زمان فراخوانی مقدار آنها به روال ارسال می شود.

بطور کلی در ویژوال بیسیک ، دو نوع روال وجود دارد:

روال *Sub* و روال *Function*

این دو روال تا حدود زیادی مشابهند و تنها تفاوت آنها این است که روال *Function* در زمان خاتمه مقداری را به فراخواننده برمی گرداند ، ولی روال *Sub* هیچ مقدار برگشتی ندارد .

روال *Sub* :

شکل کلی ایجاد یک روال *Sub* بصورت زیر است :
کد:

```
Sub    Routine_name ( Argument)
Statements
End Sub
```

در این ساختار *Routine_name* نام روال می باشد . سعی کنید اسامی انتخاب شده متناسب با عملی باشد که روال انجام خواهد داد . باید تمام آرگومانهای مورد نیاز روال در بین دو پرانتز تعریف شود ، حتی اگر هیچ آرگومانی مورد نیاز نباشد وجود دو علامت پرانتز لازم است
نکته : نمی توان در داخل یک روال از متغیری با نام روال استفاده کرد
با هربار فراخوانی روال دستورات آن از اولین دستور تا دستور *End Sub* و یا رسیدن به دستور *Exit Sub* (درصورت وجود) ، اجرا خواهد شد و پس از آن کنترل برنامه به خط بعد از فراخوانی روال برخواهد گشت.

روال *Function* :

در مواردیکه نیاز به دریافت یک نتیجه ، بعد از اجرای یک روال وجود دارد .باید از روال های *Function* استفاده کرد . شکل کلی این نوع روال ، بصورت زیر است :
کد:

```
Function   Function_Name ( Argumen)
Statements
End Function
```

مشاهده می کنید که روال *Function* تا حدود زیادی مشابه روال *Sub* می باشد . ولی روال *Function* باید مقداری را برگرداند . پس دانستن دو مطلب ضروری است ، اول اینکه مقدار برگشتی چگونه باید دریافت شود ، و دیگر اینکه چگونه یک مقدار از داخل تابع ارسال می شود .
مقدار برگشتی تابع را می توان به صورتهای زیر دریافت کرده ، مورد استفاده قرار داد .
تابع را در سمت راست یک عبارت جایگزینی قرار می دهیم .
از تابع در سمت راست یک عبارت محاسباتی استفاده می کنیم .
نتیجه تابع را به عنوان آرگومان یک روال دیگر به داخل روال ارسال می کنیم.

نکته ها:

توجه کنید که هنگام فراخوانی روال باید تعداد مقادیر ارسالی باتعداد آرگومانهای روال مساوی باشد . طریقه ارسال یک مقدار از داخل تابع نیز به راحتی انجام میشود . به این ترتیب که در داخل روال از نام تابع ، بعنوان یک متغیر استفاده می کنیم و در زمان برگشت از تابع به دستور فراخواننده، مقدار موجود در این متغیر به دستور فراخواننده ارسال می شود .

نکته : اگر هیچ مقداری در متغیر نام تابع قرار ندهیم مقدار صفر را برمی گرداند .

توجه کنید که می توان در داخل روالها ، روالهایی را نیز فراخوانی کرد . ولی باید فراموش نکنید که هنگام بازگشت از هر روال ، کنترل برنامه به دستور بعد از دستور فراخوانی خواهد رفت . به عنوان نکته آخر باید بدانید که در تعریف روالهایی از نوع *Function* می توان نوع متغیر برگشتی را تعیین نمود.

مثال:

Function Area (S1 As Single , S2 As Single) As Single

ساختار کنترلی : (If/Then/Else)

استفاده از این دستور زمانی مفید است که بخواهیم از بین دو حالت ممکن یکی را انتخاب نماییم ، کاربرد آن به صورت زیر می باشد

IF شرط مورد ارزیابی *Then* دستور ۱ *Else* دستو ر ۲

نحوه کار بدین صورت است که پس از ارزیابی شرط منطقی ، اگر حاصل درست (*True*) بود دستور ۱ اجرا می شود و دستور ۲ اجرا نخواهد شد ولی چنانچه حاصل شرط نادرست(*False*) باشد دستور ۲ اجرا می گردد و دستور ۱ اجرا نخواهد شد.

نکته ۱ : دستورات ۱و ۲ نیز می توانند به صورت یک بلوک نیز باشند ، یعنی بجای اجرای فقط یک دستور می توان گفت در صورت برقرای شرط یا بلعکس ، مجموعه ای از دستورات اجرا شوند

```
IF شرط مورد ارزیابی Then
……
……
Else
……
……
End if
```

```
IF شرط مورد ارزیابی Then
دستور ۱
Else
دستو ر ۲
End if
```

نکته ۲ : استفاده از قسمت *Else* اختیاری است .

دستور *If* تو در تو : اگر در بررسی شرط بیش از دوحالت قابل انتخاب داشتیم بایستی از *If*های تو در تو استفاده نماییم .

مثال : برنامه ای بنویسید که عددی را از ورودی دریافت نماید و تشخیص دهد که عدد مثبت ، منفی یا صفر است :

```
N=inputbox("Enter Number :")
If n>0 then
Print ('Positive')
Else
If n<0 then
Print ('Negetiv')
Else
Print ('Zero')
End if
```

مثال: می خواهیم دستورالعملهایی بنویسیم که اگر مقدار ذخیره شده در متغیر *k* زوج باشد پیغام *"Even"* و در غیر ایـن صـورت (فـرد بـودن متغیر) پیغام *"odd"* را چاپ نماید.

```
If  k mod 2=0 then
    Print "Even"
Else
    Print  "odd"
End if
```

Else If : وقتی تعداد انتخاب ها زیاد یاشد (بیش از ۳ انتخاب) کنترل *If* های تو در تو چندان ساده نیست بنابراین از ساختار *Else If* استفاده می کنیم :

تمرین : برنامه ای بنویسید که نمره دانشجویی را از ۱۰۰ نمره دریافت و طبق جدول زیر رتبه آن را چاپ نمایید :

A	نمره > = *90*
B	*90* > نمره ≥*80*
C	*80* > نمره ≥*70*
D	*70* > نمره ≥*60*
F	*60* > نمره

```
N=inputbox ("Enter Grade :")
If Grade >= 90 then
   Print ('A')
Else if Grade >= 80 then
   Print ('B')
Else if Grade >=70 then
   Print ('C')
Else if Grade >=60 then
  Print ('D')
Else
  Print ('F');
End if
```

دستور شرطی *IIF*:

***IIf(*(دستور ۲,دستور ۱ ,عبارت منطقی (شرط))**

در تابع فوق عبارت منطقی مورد ارزیابی قرار می گیرد اگر ارزش آن درست بود دستور ۱ اجرا می شود در غیر اینصورت دستور ۲ اجرا می شود .

مثال : کد زیر باعث تشخیص زوج یا فرد بودن عدد *n* می شود.

```
X=IIf(n mod 2= 0 , "Even" , "Odd")
Print x
```

دستور *Case* : با استفاده از دستور *Case* شما قادر به بررسی شرط یا شرایط خاصی خواهید بود . در حقیقت عملکرد این دستور معادل استفده از چندین دستور *If* است . شکل کلی بکار گیری این دستور به صورت زیر است :

عبارت مورد ارزیابی *Select Case*
دستور۱ : مقدار۱ *Case*
دستور۲ : مقدار *Case 2*
.
.
Case Else
دستور
End Select

توجه : مقدار ها می توانند به صورت تک مقداری یا چند مقدار (با استفاده از کاما) یا محدوده ای باشند .

نکته ۱: عبارتی که مورد ارزیابی قرار می گیرد و همچنین مقادیر استفاده شده ، حتماً باید از نوع ترتیبی باشند . استفاده از نوع اعشاری غیر مجاز است .

نکته ۲ : موارد بکار رفته نباید تکراری باشند .

نکته ۳ : اگر مقادیر انتخابی مربوط به یک مورد پراکنده بود آنها را با کاما (,) از یکدیگر جدا می کنیم

نکته ۴ : اگر مقادیر انتخابی مربوط به یک مورد به دنبال هم و پیوسته بود می توانیم آنها را به صورت زیر قلمرو و با استفاده از علامت (..) مشخص نماییم .

نکته ۵ : وجود قسمت *Else* اختیاری است .

نکته ۶ : عبارت مقابل *Case* و مقادیر استفاده شده باید هم نوع باشند .

نکته ۷ : هر کدام از دستورات می توانند یک بلوک باشند .

نکته ۸ : عبارت مورد ارزیابی می تواند به صورت یک عمل محاسباتی یا منطقی باشد .

نکته ۹ : نوع *String* یک نوع ترتیبی نیست و نمی توان آنرا مقابل عبارت *case* قرار داد .

در این ساختار می توان به تعداد دلخواه و بسته به شرایط برنامه مورد نظر ، شرط را توسط تعیین مقدار (در مقابل دستورات *Case* استفاده کرده، به ازاء هر مورد بلوک دستورات مربوطه *(Statements)*را جهت اجرا قرار داد. در پایان دستورات به اختیار می توان از دستور *Case else*نیز استفاده کرد، که در این صورت اگر هیچ کدام از مقادیر موجود در *Case* ها با مقدار عبارت مورد ارزیابی *(Expression)* ساختار برابر نباشد ، دستورات بعد از *Case else* اجرا می شود.

یک نکته در مورد ساختار: *Select Case*

در ساختار *Select case* می توان بجای استفاده از *Expression match* ، از عملگرهای *is* و *To*نیز استفاده نمود.

به مثال زیر توجه کنید:
کد:

```
Select Case no
        Case is < 10
                Print "Flunked"
        Case 10 to 19
                Print "Accepted"
        Case 20
                Print " Accepted & Awarded "
        Case else
                Print "No. is invalid"
End select
```

در مثال بالا اگر مقدار *no* کمتر از ۱۰ باشد پیغامی با عنوان *Flunked* و اگر بین ۱۰ تا ۱۹ باشد پیغام *Accepted*و اگر برابر با مقدار ۲۰ باشد پیغام *Accepted&Awarded*صادر خواهد شد.

آشنایی با حلقه های تکرار :

در بسیاری از موارد لازم است که یک یا چند دستورالعمل به دفعات معین و یا وابسته به شرطی ، تکرار شوند . از این رو حلقه های مختلفی در *VB* وجود دارند که با آنها آشنا می شویم . درکل حلقه ها به دو دسته : معین و نامعین تقسیم می شوند.

حلقه های معین : تعداد دفعات انجام آنها مشخص می باشد . (*For*)

حلقه های نامعین: تعداد دفعات آنها نا مشخص و وابسته به شرط می باشد . که این نوع حلقه ها نیز به دو دسته مثبت و منفی تقسیم میشوند. (*Do loop*) نکته : حلقه (*Do loop*) یک حلقه تکرار نامحدود است

منظور از حلقه تکرار شرطی مثبت : انجام عملیات تا هنگامی که شرط برقراراست .*(Do While)* یا (*While Wend*)

منظور از حلقه تکرار شرطی منفی : انجام عملیات تا هنگامی که شرط برقرارنیست. *(Do Until)*

حلقه تکرار معین (*For*) :

این دستور با استفاده از یک شمارنده (*Counter*) ، یک دستور یا بلوکی از دستورات را به تعداد دلخواه ما تکرار می کند .

از این حلقه به دو صورت افزایشی یا کاهشی می توان استفاده کرد .

شکل کلی دستور به صورت زیر است :

For =نام شمارنده مقدار اولیه *To* مقدار نهائی *[Step* میزان پرش *]*

دستوریا دستورات

Next [نام شمارنده*]*

مثال :

```
lname=inputbox ('Enter Last name:')
For I:= 1 to 10 do
 Print (lname);
Next I
```

```
For I:= 1 to 10 do
Print (i);
Next I
```

```
For I:= 10 to  Step -1
Print (i);
Next I
```

نکته ۱ : در حلقه *For* افزایشی برای اینکه دستورات بدنه حلقه ، حداقل یک بار تکرار شود ، مقدار اولیه باید کوچکتر یا مساوی مقدار نهائی باشد . اما درحلقه *For* کاهشی مقدار اولیه باید بزرگتر یا مساوی مقدار نهائی باشد و میزان پرش حتماً منفی انتخاب شود . چون پیش فرض ۱+ می باشد .

نکته ۲ : در حلقه *For* شرط حلقه ابتدای ورود به حلقه مورد بررسی قرار میگیرد .

نکته ۳ : متغیر حلقه حتماً باید از نوع ترتیبی باشد (عددی صحیح ، کاراکتری ، منطقی) .استفاده از متغیر نوع *Single* یا *String* مجاز نیست .

نکته ۴ : مقدار اولیه و نهایی باید با یکدگر سازگاری داشته باشند .

نکته ۵ : بهتر است مقدار متغیر حلقه *For* در ، درون حلقه تغییر داده نشود ، البته با تغییر متغیر درون حلقه پیغام خطایی از مفسر داده نمی شود ولی ممکن است اجرای برنامه دچار مشکل شود .

نکته ۶ : تغییر مقدار اولیه یا نهائی ، درون حلقه *For* هیچ تأثیری بر روند اجرای برنامه ندارد .

نکته ۷ : تعداد تکرار حلقه *For* از فرمول زیر محاسبه می شود .

در حلقه *For* افزایشی = (مقدار نهائی – مقدار اولیه) + ۱

در حلقه *For* کاهشی = (مقدار اولیه – مقدار نهائی) + ۱

نکته ۸ : برای محاسبه تعداد دفعات تکرار در حلقهای تکرار تو در تو : تعداد دفعات هریک را جدا گانه طبق نکته قبل محاسبه و سپس حاصل را در یکدگر ضرب کنید .

خروج اضطراری از حلقه *For* به کمک دستور *Exit for* :

برخی مواقع ممکن است لازم باشد تا بسته به یک شرط خاص ، کار حلقه پیش از موئد خاص پایان پذیرد . در این صورت از دستور *Exit For* برای متوقف کردن حلقه استفاده می کنیم و کنترل برنامه به سطر بعد از انتهای حلقه منتقل می شود .

حلقه تکرار شرطی مثبت (*While*) :

هرگاه بخواهیم دستور یا دستوراتی تا برقرار بودن شرطی خاص تکرار شوند از این نوع حلقه استفاده می کنیم که شکل کلی آن به صورت زیر است :

نوع دوم

Do
دستور یا دستورات
Loop While **شرط مورد ارزیابی**

نوع اول

Do While **شرط مورد ارزیابی**
دستور یا دستورات
loop

حلقه تکرار شرطی منفی (*Do Until*) :

هر گاه بخواهیم انجام عملیات تا هنگامی که شرط برقرارنیست ادامه یابد. از دستور حلقه سازی *Do Until* استفاده می نماییم و تفاوت آن با حلقه *Do While* در این است که تا زمانی که یک شرط منطقی خاص برقرار نشده است حلقه ادامه دارد . شکل کلی دستور به صورت زیر می باشد :

نوع دوم

Do
دستور یا دستورات
Loop Until **شرط مورد ارزیابی**

نوع اول

Do Until **شرط مورد ارزیابی**
دستور یا دستورات
loop

نحوه کار نوع دوم در هر دو نوع حلقه *While , Until* : بدین صورت است که ابتدا یکبار بدون بررسی شرط ، دستورات حلقه انجام انجام می شود سپس شرط یا عبارت منطقی ارزیلبی می گردد . اگر نتیجه نادرست باشد حلقه یکبار دیگر تکرار می شود و اگر شرط درست باشد حلقه پایان می یابد .

نکته ۱ : در حلقه *While* اجرای حلقه منوط به برقراری شرط است ولی در حلقه *Untile* اجرای حلقه منوط به برقرار نبودن شرط است . یا به عبارتی در حلقه های نوشته شده توسط *While* تا وقتی مقدار شرط درست باشد حلقه اجرا خواهد شد ولی در حلقه های نوشته شده با *Until* با محقق شدن شرط اجرا تمام خواهد شد.

نکته ۲ : در دستورات **نوع اول** شرط حلقه در همان ابتدای ورود به حلقه بررسی می شود ولی در **نوع دوم** دستورات درون حلقه یکبار اجرا شده سپس شرط مورد ارزیابی قرار می گیرد .

نکته ۳ : حلقه های متداخل باید به صورتی باشد که یکی کاملاً درون دیگری قرار بگیرد ، ندون آنکه یکدیگر را قطع نمایند .

نکته ۴ : در حلقه های متداخل ابتدا بیرونی ترین حلقه ، اولین دور خود را آغاز می کند ولی از بقیه حلقه ها دیر تر پایان می یابد .

نکته ۵ : برای محاسبه تعداد دفعات تکرار در حلقهای تکرار تو در تو : تعداد دفعات هریک را جدا گانه طبق نکته قبل محاسبه و سپس حاصل را در یکدگر ضرب کنید .

نکته ۶: شرط در حلقه می تواند هر عبارت یا متغیر منطقی باشد و مشخص کننده مقدار *False* یا *True* می باشد.

نکته ۷ :متغیری که به عنوان شرط آزمایش می شود باید در داخل حلقه تغییر یابد و این تغییر باید طوری باشد که حلقه نقطه پایانی داشته باشد.

به مثال زیر توجه کنید :

کد:

```
Dim I as integer
 I=1
Do while I<=5
 I=I+1      (نکته ۷)
Loop
```

حلقه این مثال به تعداد ۵ بار اجرا می شود و I به هنگام خروج از حلقه برابر ۶ است.

نکته ۸ : از دستور *Exit Do* همانند *Exit For* می توان برای خروج از حلقه استفاده کرد .

جعبه کنترل *CommonDialog* :

جعبه گفتگو ها در ویژوال بیسیک باعث می شود برنامه های شما پیشرفته تر شود و کاربران از کار کردن با برنامه شما لذت ببرند.

جعبه گفتگو ها در ویژوال بیسیک به شش(۶) دسته تقسیم میشوند که عبارتند از:

- **جعبه گفتگوی انتخاب رنگ** : گفتگویی که به کاربر امکان انتخاب رنگ مورد نظر و حتی دستکاری آن ها رامی دهد.
- **انتخاب فونت:** گفتگویی برای انتخاب فونت ونوع و اندازه و سبک آنها.
- **باز کردن فایل:** گفتگویی برای باز کردن فایل از پوشه ها و درایوها حتی درایو های شبکه.
- **سیو(ذخیره کردن) فایل:** گفتگویی برای ذخیره کردن فایل در پوشه ها و درایوها.
- **چاپ:** گفتگویی برای انتخاب چاپگر و سایر تنظیمات آن.
- **کمک ویندوز:** سیستم کمک ویندوز را فعال میکند و اگر برنامه شما دارای کمک باشد امکان استفاده از آن را برای کاربران فراهم می آورد.

برای اضافه کردن کنترل جعبه گفتگو به برنامه خود باید:

۱ -بازدن کلیدهای ترکیبی *Ctrl + T* یا از منوی *Project* گزینه *Components* را انتخاب و گفتگوی *Components* را باز کنید.

۲ -کنترل *Microsoft Common Dialog Control 6.0* را پیدا کنید و آن را تیک بزنید.

۳ -دکمه *OK* را بزنید تا این کنترل به جعبه ابزارتان اضافه شود.

متدهای کنترل جعبه گفتگو:متدهای کنترل جعبه گفتگو عبارتند از:

۱ *Show Open* -گفتگوی باز کردن فایل را نمایش خواهد داد.
۲ *Show Save* -گفتگوی ذخیره کردن فایل را نمایش میدهد.
۳ *Show Printer* -گفتگوی انتخاب چاپگر را نمایش میدهد.
۴ *Show Color* -گفتگوی انتخاب رنگ را نمایش میدهد.
۵ *Show Font* -گفتگوی انتخاب فونت را نمایش میدهد.
۶ *Show Help* -گفتگوی کمک ویندوز را نمایش میدهد.

برای درست کردن اینکه کنترل ما کدام گفتگو را نمایش دهد به صورت زیر عمل میکنیم:

CommonDialog1.ShowOpen
یا
CommonDialog1.ShowSave
یا
CommonDialog1.ShowFont

یک مثال برای درک کردن کار با جعبه گفت و گو:

ابتدا یک کنترل جعبه گفتگو با نام *CommonDialog1* روی فرم و یک *Text Box* و یک دکمه فرمان به نام *Command1* روی فرم

خود قرار دهید روی دکمه *Command1* خود دوبار کلیک کنید تا پنجره کد باز شود.
سپس:

کدهای زیر را در روال رویداد *Command1_Click* بنویسید.

```
Private Sub Command1_Click
CommonDialog1.ShowColor
Text1.BackColor = CommonDialog1.Color
End Sub
```

کار برنامه:
در خط دوم ما گفتیم که کنترل جعبه گفتگو گفتگوی رنگ را باز کند.
در خط سوم نیز گفتیم که رنگ پس زمینه *Text box*را برابر رنگ انتخابی جعبه گفتگوی رنگ کند.

ساخت منو:

- برای قراردادن و طراحی منو برای فرم از منوی(*tools → menu editor*)را انتخاب کنید.
- برای افزودن گزینه ای جدید به منو عنوان آن را در کادر *caption* و نام آن را در کادر *name* تایپ کنید .
- برای آنکه گزینه فعلی زیر منوی گزینه قبلی شود کلید جهت راست را فشرده و برای لغو آن از کلید جهت سمت چپ استفاده کنید .
- برای ترتیب گزینه ها در منو از فلش بالا و پایین استفاده میکنیم و برای آنکه کنار گزینه علامت تیک گذاشته شود از کادر *checked*برای فعال کردن گزینه از کادر *enabled* و برای قابل رویت بودن گزینه از کادر *vizible* استفاده میشود .
- برای استفاده کردن گزینه ای بین گزینه های دیگر از *insert* و برای حذف یک گزینه از *delete* استفاده میشود.
- برای آنکه بین گزینه های منو خطی قرار گیرد که مثلا بعضی گزینه ها را از بعضی دیگر جدا کند , گزینه ای با *caption* خط فاصله ایجاد میکنیم و برای قرار دادن خط زیر برای یکی از حروف گزینه به منظور انتخاب گزینه به کمک *alt* قبل از حرف مورد نظر در *caption* کاراکتر& را قرار میدهیم : مثال *&file :*

خاصیت : *enabled* در حین اجرای برنامه تغیر این خاصیت به *true* و *false* باعث غیر فعال شدن گزینه میشود .
خاصیت : *visible* اختصاص مقدار *false* به این خاصیت گزینه را پنهان میکند .
خاصیت : *checked* خاصیت مقدار *true* به این خاصیت باعث میشود تا کنار گزینه علامت تیک ظاهر شود.
حالا شما می توانید به راحتی یک منو در *VB6* بسازید.

کنترلهای *Check Box* و *Option Button*

از زمانی که با ویندوز کار می کنید، از کنترلهای *Check Box* و *Option Box* استفاده بسیاری برده اید.اگر بخواهیم به کاربر *(User)* این امکان را بدهیم که از بین چند گزینه، یک ، دو یا چند مورد را انتخاب کند و یا حتی هیچ یک را انتخاب نکند از کنترل *Check Box* و اگر بخواهیم کاربر، از میان چند گزینه، یک و فقط یک گزینه را انتخاب کند، از کنترل(*Option Button* کلیدهای رادیویی) *استفاده میکنیم.*

این دو کنترل مشخصه ای به نام *Value* دارند که از صفر یا یک بودن مقدار آن، می توانیم به انتخاب نشده بودن و انتخاب شده بودن آن گزینه خاص، پی ببریم. لذا اگر برروی یک فرم چند *Check Box* وجود داشت، می تواند *Value* هر یک از آنها صفر یا یک باشد، اما اگر برروی فرمی، چند *Option Button* وجود داشت، تنها یکی از آنها می تواند *Value* برابر یک داشته باشد و وقتی کاربر، یکی دیگر را انتخاب می کند، به طور خودکار، *Value*قبلی صفر و *Value* کنترل انتخاب شده، یک می شود.

*VB*دارای دو ثابت *(Constant)* به نامهای *VBChecked* و *VBUnchecked* می باشد که به ترتیب برابر یک و صفر می باشند که می توانیم مقدار *Value* کنترلهای ذکر شده را با این ثوابت مقایسه کنیم.

کنترل *Check Box* ، علاوه بر دو مقدار صفر و یک برای مشخصه *Value* ، مقدار برابر ۲ نیز برای آن دارد که اگر به آن *Set* شود، این کنترل، خاکستری رنگ *(Gray)* می شود که معمولاً در ویندوز نشان از پیش فرض بودن دارد. همچنین کنترل مذکور، مشخصه ای هم به نام *Style*دارد که اگر مقدار آن را از صفر به یک تغییر دهیم، شکل *Check Box* به شکل دکمه ای در می آید که یک بودن مقدار*Value* ، باعث فشرده شدن کلید و صفر بودن آن باعث بالا بودن کلید می*شود.

کار با فایل هایی از نوع *Excel* و*Word*

به دلیل این که برنامه *Excel* و *Word* یکی از ساده ترین و پرکاربردترین برنامه های مورد استفاده بوده و ساخت شرکت مایکروسافت است و ویژوال بیسیک به راحتی قادر به برقراری ارتباط با آن است ، می خواهم به صورت خلاصه روش ایجاد یک فایل از نوع *Excel* و *Word* و برقراری ارتباط با آنها را توضیح دهیم .
برای این که شما از امکانات کار با *Excel* و *Word* در ویژوال بیسیک *6* برخوردار باشید باید از رابطی که مخصوص خود *Excel* و *Word* است استفاده نمایید .
برای این کار شما باید از دیالوگ *References* گزینه های زیر را تیک بزنید:

Microsoft Excel 9.0 Object Library
Microsoft Word 9.0 Object Library

البته بسته به نسخه آفیسی که شما بر روی سیستم خود نصب می کنید شماره یا ورژن این گزینه ها تغییر خواهد کرد.

کار با : *Excel*

در ابتدا شما باید تعریفات زیر را در قسمت *General* برنامه خود انجام دهید :

```
Dim X_Excel As Excel.Application
Dim X_WorkBook As Excel.Workbook
Dim X_WorkSheet As Excel.Worksheet
```

حالا برای ایجاد یک فایل از نوع *Excel* از دستور زیر باید استفاده کنید:

```
Set X_Excel = New Excel.Application
```

سپس یک صفحه کاری ایجاد کنید که باید از دستور زیر استفاده کنید:

```
Set X_WorkBook = X_Excel.Workbooks.Add
```

پس از این کار شیت *(Sheet)* مورد نظر را باید انتخاب کنید:

```
(Set X_WorkSheet = X_WorkBook.Worksheets(1
```

که در این مثال *Sheet1* را انتخاب کردیم.
همانطور که می دانید صفحه *Excel* به صورت گسترده بوده و کار با آن بر اساس خانه های موجود در آن است که هر کدام آدرس مخصوصی دارد.

برای مثال خانه شماره یک دارای آدرس (۱ , ۱) می باشد . ما هم از این آدرسها برای دسترسی به این خانه ها استفاده می کنیم:

داده مورد نظر = (ستون , سطر*X_WorkSheet.Cells*)

```
X_WorkSheet.Cells( 1 , 1 ) = "VB"
X_WorkSheet.Cells( 1 , 2 ) = "Bala"
```

در مثال بالا رشته *VB* در اولین خانه و رشته *Bala* در خانه دوم ردیف اول صفحه یا جدول ذخیره می شود و محدوتی برای وارد کردن انواع داده وجود ندارد و شما می توانید هر نوع داده ای را در خانه ها قرار دهید.

پس از ایجاد یک صفحه کاری و قرار دادن داده در خانه های آن نوبت به ذخیره کردن آن به عنوان یک فایل *Excel* می رسد که باید از دستور زیر برای این کار استفاده کنید:

X_WorkBook.SaveAs FileName:= "C:\Smple.xls"

همچنین شما میتوانید برای نمایش صفحه کاری خود توسط برنامه اکسل به صورت زیر عمل کنید:

X_Excel.Visible = True

و برای خروج از صفحه کاری از دستور زیر استفاده کنید:

X_Excel.Quit

این دستور زمانی کاربرد دارد که از دستور قبلی آن استفاده نمایید.

کار با : *Word*

کار با این برنامه هم مانند کار با *Excel* است و تفاوت آنها بیشتر در قسمت وارد کردن داده ها و اطلاعات می باشد . مانند قبل تعریفات زیر را در قسمت *General* قرار دهید:

Dim X_Word As Word.Application
Dim X_Doc As Word.Document

حال برای ایجاد یا باز کردن یک فایل از نوع *Word* دستور زیر را به کار ببرید:

Set X_Word = New Word.Application

سپس شما باید یک پرونده *(Document)* جدید ایجاد کنید:

Set X_Doc = X_Word.Documents.Add

پس از انجا این کارها نوبت به وارد کردن داده ها و اطلاعات است که برای این کار شما بیشتر باید از متد *Selection* مربوط به *X_Word* استفاده نمایید:

X_Word.Selection.Borders.OutsideLineStyle = wdLineStyleInset
X_Word.Selection.ParagraphFormat.Alignment = wdAlignParagraphCenter
X_Word.Selection.Font.Bold = True
X_Word.Selection.Font.Size = 20
X_Word.Selection.Text = "VB Is For All"

در این مثال پس از تنظیمات دلخواه در صفحه پرونده نوشته ای را در آن قرار می دهیم . البته این متدها فقط تعداد اندکی از امکانات کار با فایل های پرونده ای است و شما باید خودتان آنها را بررسی کنید.

حال مانند مثال قبل نوبت به ذخیره کردن پرونده کاری می رسد که روش آن مانند مثال قبل است:

"X_Doc.SaveAs FileName:="C:\Sample.Doc

برای نمایش پرونده کاری خود توسط برنامه *Word* دستور زیر استفاده کنید:

X_Word.Visible = True

برای درک مطلب به مثال زیر مراجعه نمایید :

```
Dim X_Excel As Excel.Application
Dim X_WorkBook As Excel.Workbook
Dim X_WorkSheet As Excel.Worksheet
Dim X_Word As Word.Application
Dim X_Doc As Word.Document

Private Sub CmdWordNew_Click()
  Set X_Word = New Word.Application
  Set X_Doc = X_Word.Documents.Add

  X_Word.Selection.Borders.OutsideLineStyle = wdLineStyleInset
  X_Word.Selection.ParagraphFormat.Alignment = wdAlignParagraphCenter
  X_Word.Selection.Font.Bold = True
  X_Word.Selection.Font.Size = ۲۰
  X_Word.Selection.Text = "This is a sample for Student"
  X_Doc.SaveAs FileName:=App.Path & "\tamrin.Doc"
  X_Word.Visible = True
End Sub

Private Sub CmdExcelNew_Click()
  Set X_Excel = New Excel.Application
  Set X_WorkBook = X_Excel.Workbooks.Add
  Set X_WorkSheet = X_WorkBook.Worksheets(۱
   X_WorkSheet.Cells(۱, ۱) = "ID"
  X_WorkSheet.Cells(۱, ۲) = "Name"
  X_WorkSheet.Cells(۱, ۳) = "Family"

  X_WorkSheet.Cells(۲, ۱) = ۱
  X_WorkSheet.Cells(۳, ۱) = ۲

  X_WorkSheet.Cells(۲, ۲) = "Hadi"
  X_WorkSheet.Cells(۳, ۲) = "Mohsen"

  X_WorkSheet.Cells(۲, ۳) = "Hamzavi"
  X_WorkSheet.Cells(۳, ۳) = "Sanati"

  X_WorkBook.SaveAs FileName:=App.Path & "\tamrin.xls"

  X_Excel.Visible = True
End Sub
```

فرمان *Shell*

فرمان *Shell* یکی از فرمان های مهم است که در *VB* پر کاربرد است. با کمک آن می توانید هر پرونده ای رو باز کنید تقریباً مثل *RUN* ویندوز. حالا چند نکته و چند دستور رو اینجا یاد آوری می کنیم :

نکات:

Shell فقط پرونده های اجرایی را اجرا می کند مثل : *EXE;COM;BAT*
که باز کردن پسوند های دیگه هم راه حل خودشونو دارن.
در قبل و بعد آدرس باید علامت کاما (") گذاشته شود.

شکل استفاده از دستور بصورت زیر است :

Shell "Comman"

در قسمت *Command* ، نام برنامه را می گذارید.

از این دستور می توانید اکثر برنامه های ویندوز را فقط با دادن نام آنها، اجرا کنید، مثل *Explorer.exe* و. *Notepad.exe*
اما برای دیگر برنامه ها باید نام و مسیر کامل آنها داده بشه.

اجرای برنامه های مربوط به کنترل پانل ویندوز :

نام پرونده های کنترل پانل دارای پسوند *cpl* هست که می خواهید اجرا کنید. این پرنده ها مربوط به خود ویندوز هستند.

برای اینکه بهتر درک کنید این آدرس رو امتحان کنید: (می توانید در داخل یک *Command Button* بنویسید).

Shell "Rundll32.exe shell32.dll,Control_RunDLL main.cpl,@0,1"

باز کردن پوشه *system32*

Shell "rundll32.exe shell32.dll,ShellExec_RunDLL"

باز کردن اضافه یا حذف برنامه ها در صفحه تنظیم:

Shell "RUNDLL32.EXE shell32.dll,Control_RunDLL appwiz.cpl"

اگه خواستید فایل های بیشتری رو باز کنید , وارد پوشه سیستم ۳۲ *(C:\Windows\System32)* شوید و در آنجا دنبال فابلهای *cpl* بگردید.

لیستی از برنامه هایی که مستقیماً با فرمان *Shell* اجرا می شوند :

Calc , Write ,Notepad ,Spider ,Winmine ,Mshearts ,freecell ,Regedit ,Taskmgr ,control fonts , control desktop
control mouse ,control keyboard ,osk ,magnify ,utilman ,mstsc ,cmd ,control admintools ,cleanmgr ,winchat
clipbrd ,dcomcnfg ,control printers ,charmap ,eudcedit ,perfmon ,control netconnections ,dxdiag ,cliconfg
sysedit ,ddeshare ,diskpart ,chkdsk ,verifier ,sigverif ,packager ,iexpress ,fsquirt , drwtsn32

به این سه دستور توجه کنید:

- *Shutdown –l –t 0*
- *Shutdown –s –t 0*
- *Shutdown –r –t 0*

شما میتونید از این دستورات برای *Log Off* ، *Restart* و یا *Shutdown* کردن ویندوز استفاده کنید. فقط کافیست که دستور دلخواه را جلوی دستور *Shell* تایپ کنید.

دستور زیر باعث *Shutdown* شدن ویندوز می شود :

Shell "Shutdown –s –t 0"

و امّا عدد صفر که آخر دستور نوشته شده مدّت زمانیست که تعیین میکند چند ثانیه بعد از اجرای دستور ، ویندوز *Shutdown* شود که در اینجا صفر قرار دادیم تا بلافاصله اینکار انجام شود.

تابع *SendKeys* :

در حالت عادی شما برای درج هر کاراکتری و یا انجام هر عملی دکمه های مورد نظرتان را ازروی کیبورد برای تایپ انتخاب می کنید.
کار این تابع همانطور که از اسمش هم پیداست این است که شما هر دکمه ای رو که می خواهید از صفحه کلید به سیستم عامل یا برنامه مورد نظر ارسال کنید را ، از طریق کد نویسی ارسال می کند.

شکل کلی تابع:

SendKeys string

مقادیری که می تواند در آرگومان *String* قرار گیرد: (کد مورد نظر حتماً باید در داخل {} قرار گیرد .)

کلید	کد کلید	کلید	کد کلید
BACKSPACE	*{BACKSPACE}, {BS}, or {BKSP}*	*`Up Arrow*	*{up}*
BREAK	*{BREAK}*	*F1*	*{f1}*
CAPS LOCK	*{CAPSLOCK}*	*F2*	*{f2}*
DEL or DELETE	*{DELETE} or {DEL}*	*F3*	*{f3}*
DOWN ARROW	*{DOWN}*	*F4*	*{f4}*
END	*{END}*	*F5*	*{f5}*
ENTER	*{ENTER}or ~*	*F6*	*{f6}*
ESC	*{ESC}*	*F7*	*{f7}*
HELP	*{HELP}*	*F8*	*{f8}*
HOME	*{HOME}*	*F9*	*{f9}*
INS or INSERT	*{INSERT} or {INS}*	*F10*	*{f10}*
LEFT ARROW	*{LEFT}*	*F11*	*{f11}*
NUM LOCK	*{NUMLOCK}*	*F12*	*{f12}*
PAGE DOWN	*{PGDN}*	*F13*	*{f13}*
PAGE UP	*{PGUP}*	*F14*	*{f14}*
PRINT SCREEN	*{PRTSC}*	*F15*	*{f15}*
RIGHT ARROW	*{RIGHT}*	*F16*	*{f16}*
SCROLL LOCK	*{SCROLLLOCK}*		
TAB	*{TAB}*		

موارد زیر مربوط به کلید های کنترلی میشود :

SHIFT +
CTRL ^
ALT %

مثال : دستور زیر کار کلید های ترکیبی *Alt+F4* رو انجام میده یعنی هر پنجره ای رو که فعال باشه می بنده و نهایتا پنجره *ShutDown* را نشان میدهد .

```
SendKeys "%{F4}"
```

مثلا این یکی یه متن را داخل یک تکست باکس *Select* می کند:

```
Text1.SetFocus
SendKeys "{Home}+{End}"
```

که در اینجا *Text1* همان تکست باکسیست که می خوایم متنش *Select* شود .

آشنایی با اشیای *FileListBox* ، *DirectoryListBox*و *DriveListBox* :

این سه شی به ترتیب جعبه لیست های فایل و فهرست و درایو می باشند. البته معمولا هیچ کدام به تنهایی کاربرد ندارند و باید میان آن ها ارتباط برقرار کرد و آن ها را به هم پیوند زد.

طرز ارتباط این سه شی:

```
Private Sub Drive1_Change()
  Dir1.Path = Drive1.Drive
 End Sub
Private Sub Dir1_Change()
  File1.Path = Dir1.Path
End Sub
```

دستور اول پیوند فهرست به درایو مربوطه و دستور دوم نسبت دهی فایل انتخابی به فولدری که فایل در آن قرار دارد می باشد.

خواص مهم: *FileListBox*

FileName نام فایل انتخاب شده در جعبه فایل.

Path مسیر فایل انتخابی.

Pattern الگو فایل را در جعبه فایل تعین می کند.

آشنایی با دستور *On Error* : این دستور در مواردی استفاده می شود که امکان رخ دادن یه خطا وجود دارد و ما می خواهیم از آن جلوگیری کنیم.

ما بعد از عبارت *On Error* از کلماتی مثل *GoTo* و *Resume* به صورت زیر استفاده کنیم مثل:

On Error Resume Next

این خط کد باعث می شود که هیچ خطایی نشان داده نشود و اگر داخل برنامه در حین اجرا خطایی رخ دهد از آن رد شده انگار نه انگار که مشکلی پیش آمده ولی در اصل برنامه دچار اشکال است ولی بروش نمیاره*!*

این خط کد باعث میشود که اگر برنامه خطا داد از آن صرف نظر کند و به خط بعد برود :

On Error GoTo line

این خط کد باعث میشود که اگه برنامه خطا داد برنامه به همان خطی که ما گفتیم (به جای *line* اسم خط رو بنویسید) می رود.

On Error GoTo 0

نکته ۱: این کدها را باید در اولبرنامه نوشته شود تا درست عمل کند .

مدیریت رشته ها در ویژوال بیسیک

توابعی که برای مدیریت رشته ها در وی بی می توانید از آنها استفاده کنید عبارتند از :

تابع *Asc* : کد اسکی اولین کاراکتر رشته ورودی را بر می گرداند.

فرمت کلی آن بصورت زیر است:

Asc(string)

تابع *Asc* کد یونیکد اولین کاراکتر را بر می گرداند.

تابع *Chr* : رشته ای را بر می گرداند که معادل کد اسکی ورودی است.

فرمت کلی آن بصورت زیر است:

Chr(charcode)

تابع *Chr* بر حسب یونیکد عمل می کند.

تابع *LCase* : تمام کاراکترهای رشته ورودی را به حروف کوچک تبدیل می کند.

فرمت کلی آن بصورت زیر است:

LCase(string)

تابع *UCase* : تمام کاراکترهای رشته ورودی را به حروف کوچک تبدیل می کند.

فرمت کلی آن بصورت زیر است:

UCase(string)

تابع *Left* : رشته ای را بر می گرداند که شامل تعداد مشخصی از کاراکترهای سمت چپ رشته ورودی است. مثال : *Left("abcdef",3)="abc"*

فرمت کلی آن بصورت زیر است:

Left(string, length)

String : ورودی رشته

Length : طول رشته مورد نظر

تابع : *Right* رشته ای را بر می گرداند که شامل تعداد مشخصی از کاراکترهای سمت راست رشته ورودی است. مثال : *Right("abcdef",3)="def"*

فرمت کلی آن بصورت زیر است:

Right(string, length)

تابع : *Space* تعداد مشخصی کاراکتر فاصله بر می گرداند.

فرمت کلی آن بصورت زیر است:

Space(number)

تابع : *Len* طول رشته ورودی را بر می گرداند . مثال :*Len("abcdefg")=7*

فرمت کلی آن بصورت زیر است:

Len(string)

تابع : *Trim* این تابع *space* هایی که در ابتدا یا انتهای رشته باشد را حذف می کند. مثال : *Trim(" abc ")="abc"*

فرمت کلی آن بصورت زیر است:

Trim(string)

توابع *LTrim* و *RTrim* فقط از چپ و راست عمل می کنند.

تابع : *Mid* این تابعی یک رشته بر می گرداند که شامل تعداد مشخصی از کاراکترهای رشته ورودی آن است . فرمت کلی آن بصورت زیر است:

Mid(string, start[, length])

string : رشته ورودی.

start :محل شروع اولین کاراکتر رشته ای که می خواهیم از رشته ورودی استخراج کنیم.

Length : این پارامتر اختیاری است و طول رشته ای است که می خواهیم از رشته ورودی استخراج کنیم . اگر این پارامتر وارد نشود کلیه کاراکترها از *start* به بعد استخراج خواهند شد.

مثال : *Mid("abcdefg",2,3)=bcd*

تابع : *Instr* این تابع محل اولین وقوع یک رشته را درون رشته دیگر نشان می دهد.

فرمت کلی آن بصورت زیر است:

InStr([start,]string1, string2[, compare])

Start این پارامتر اختیاری است و محل شروع جستجو را نشان می دهد . اگر این پارامتر وارد نشود جستجو از ابتدای رشته آغاز می شود.

String1 رشته ای که جستجو در آن انجام می شود.

String2 رشته مورد جستجو

Compare این پارامتر اختیاری است و نوع جستجو را نشان می دهد . اگر این پارامتر ۰ داده شود جستجوی متنی انجام می شود و اگر ۱ داده شود جستجوی باینری انجام می شود.

مثال : *Instr(3,"abcdabg","ab")=5*

اگر طول رشته *string1* برابر صفر باشد مقدار بازگشتی صفر است . اگر *string1* یا *string2* برابر *Null* باشد مقدار بازگشتی نیز *Null* است . اگر طول رشته *string2* برابر صفر باشد مقدار بازگشتی *start* خواهد بود . اگر رشته *string2* درون *string1* پیدا نشود مقدار بازگشتی صفر است . اگر *start* بزرگتر از طول رشته *string1* باشد مقدار بازگشتی صفر است.

تابع : *InstrRev* برعکس تابع *Instr* می باشد یعنی عمل جستجو را از انتهای رشته انجام می دهد. فرمت کلی آن بصورت زیر است:

InstrRev(stringcheck, stringmatch[, start[, compare]])

تابع : *Replace* رشته ای را برمی گرداند که در آن یک رشته خاص با رشته دیگری به تعداد دفعات مشخصی جایگزین شده است. فرمت کلی آن بصورت زیر است :

Replace(expression, find, replace[, start[, count[, compare]]])

Expression : رشته اصلی
Find : رشته مورد جستجو
Replace : رشته جایگزین
Start : محل شروع جایگزینی . در صورتیکه این متغیر وارد نشود جایگزینی از ابتدا رشته انجام می شود.
Count : تعداد دفعات جایگزینی . در صورتیکه این متغیر وارد نشود جایگزینی در تمام رشته انجام خواهد شد.
Compare : نوع جستجو را نشان می دهد . اگر این پارامتر ۰ داده شود جستجوی متنی انجام می شود و اگر ۱ داده شود جستجوی باینری انجام می شود.

مثال : *Replace(“abcadea”,”a”,”x”)=”xbcxdex”*

اگر طول رشته ***expression*** برابر صفر باشد مقدار بازگشتی رشته ای با طول صفر است . اگر طول رشته ***find*** صفر باشد مقدار بازگشتی خود ***expression*** است . اگر طول رشته ***replace*** صفر باشد مقدار بازگشتی ***expression*** ای است که در آن تمام ***find*** ها حذف شده است . اگر ***start*** بزرگتر از طول رشته ***expression*** باشد مقدار بازگشتی رشته ای با طول صفر است . اگر ***count*** برابر صفر باشد مقدار بازگشتی خود ***expression*** است.

تابع : *StrReverse* رشته ای را برمی گرداند که کاراکترهای آن به ترتیب عکس کاراکترهای رشته ورودی است.
فرمت کلی آن بصورت زیر می باشد:

StrReverse(expression)

مثال : *StrReverse(“abcd”)=”dcba”*

تابع : *Split* آرایه ای از تعداد مشخصی رشته برمی گرداند که این رشته ها توسط یک کاراکتر جداکننده (*delimiter*) از درون یک رشته استخراج شده اند.
فرمت کلی آن بصورت زیر است:

Split(expression[, delimiter[, limit[, compare]]])

Expression : رشته اصلی
Delimiter : این پارامتر اختیاری است و کاراکتر جداسازی را نشان می دهد . در صورتیکه این پارامتر وارد نشود کاراکتر فاصله (" ") برای جداسازی استفاده می شود . در صورتیکه طول این کاراکتر صفر باشد یک آرایه تک عضوی که شامل کل *expression* است برگردانده می شود.
Limit : تعداد رشته های موجود در آرایه را نشان می دهد . در صورتیکه این پارامتر داده نشود کلیه رشته های جداشده در آرایه خروجی قرار می گیرند.
Compare : نوع جستجو را نشان می دهد . اگر این پارامتر ۰ داده شود جستجوی متنی انجام می شود و اگر ۱ داده شود جستجوی باینری انجام می شود.

مثال :
Dim Ar(3) as String
Ar=Split("a#bd#cde","#")

تابع : *Join* تعدادی رشته موجود در یک آرایه را بهم متصل می کند و رشته حاصل شده را بعنوان نتیجه بر می گرداند.
فرمت کلی آن بصورت زیر است:

Join(sourcearray[, delimiter])

Sourcearray : آرایه شامل رشته هایی که می خواهیم بهم متصل کنیم.
Delimiter : کاراکتری که برای اتصال رشته ها بهم استفاده می شود . این کاراکتر در بین رشته اهی اتصالی می آید و اگر داده نشود از کاراکتر فاصله استفاده می شود . اگر طول این کاراکتر صفر باشد رشته های بدون هیچ جداکننده ای بهم متصل می شوند.

مثال :
Dim Ar(3) as String
Ar(1)="ab"
Ar(2)="c"
Ar(3)="def"
Join(Ar,"")="ab*c*def"*

تابع : *StrComp* این تابع دو رشته ورودی را با هم مقایسه می کند.
فرمت کلی این تابع بصورت زیر است:

StrComp(string1, string2[, compare])

String1 : رشته اول
String2 : رشته دوم
Compare : نوع مقایسه را نشان می دهد . اگر این پارامتر ۰ داده شود مقایسه متنی انجام می شود و اگر ۱ داده شود مقایسه باینری انجام می شود.
اگر *string1* کوچکتر از *string2* باشد مقدار بازگشتی ۱- است . اگر دو رشته مساوی باشند مقدار بازگشتی صفر است . اگر *string1* بزرگتر از *string2* باشد مقدار بازگشتی ۱ است.

تابع : *StrConv* در یک رشته ورودی تغییراتی را اعمال می کند.
فرمت کلی آن بصورت زیر است:

StrConv(string, conversion)
String : رشته ورودی
Conversion : نوع عمل تبدیل را نشان می دهد . مقادیر ممکن این متغیر عبارتند از:

مقدار	توضیح
۱	تبدیل به حروف بزرگ
۲	تبدیل به حروف کوچک
۳	تبدیل اولین کاراکتر هر لغت در رشته به حرف بزرگ
۶۴	تبدیل به یک رشته یونیکد
۱۲۸	تبدیل از رشته یونیکد به کدپیج پیش فرض سیستم

این تابع یکی از کامل ترین تابع ها در *vb* است.

آشنایی با توابع تاریخ و زمان :

تابع *Date*

این تابع یک مقدار تاریخی را که نشان دهنده تاریخ سیستم می باشد را بازمی گرداند . این تابع فاقد آرگومان است و شکل کلی آن به صورت زیر است *Date*

Print Date : باعث چاپ شدن تاریخ جاری سیستم می شود .

نکته : با تابع *Date* می توانید تاریخ سیستم را تنظیم کنید . برای این کار می توانید از این تابع به صورت زیر استفاده کنید:
Date = تاریخ

تابع *Day*

این تابع با دریافت یک مقدار تاریخی ، یک عدد صحیح بین ۱ تا ۳۱ را که نشانگر عدد روز است بازمی گرداند . شکل کلی این تابع به صورت زیر است:

Day (date)

آرگومان *Date* می تواند از نوع *Variant* ، عبارت عددی ، عبارت رشته ای و یا ترکیبی از آنها باشد که بیانگر تاریخ معینی است.

تابع *Datediff*

به وسیله این تابع می توانید فاصله زمانی بین دو تاریخ معین را بر اساس روز ، هفته ، و یا ماه و غیره ،*تعیین کند . مقدار بازگشتی این تابع از نوع *Long* می باشد.

شکل کلی این تابع به صورت زیر می باشد:

DATEDIFF (interval, date1, date2 [, firstdayofweek [, firstweekofyear]])

این تابع دارای سه آرگومان اجباری و دو آرگومان اختیاری است .

آرگومان *interval* فاصله زمانی را بر اساس یکی از مقادیری که بیان خواهد شد معین می کند . این آرگومان یک عبارت رشته ای است.
آرگومان های *date1,date2* که می تواند از نوع تاریخی یا *Variant* باشند ، تابع فاصله زمانی بین این دو آرگومان را حساب می کند.
آرگومان های چهارم و پنجم به ترتیب اولین روز هفته که در صورت تعیین نشدن به صورت پیش فرض یکشنبه است و بعدی ولین هفته سال است که مقدار پیش فرض آن اولین هفته ماه *January* است.
مقادیر آرگومان *interval* به شرح زیر هستند:
عبارت رشته ای *"yyyy"* که مقدار بازگشتی آن تعداد سال است ، *"q"* که تعداد فصل ها را بازگشت می دهد ، عبارت *"m"* معین کننده تعداد ماه ، *"y"*و *"d"* تعداد روز ، *"w"* و *"ww"* تعداد هفته ، *"h"* تعداد ساعت ، *"n"*مقدار به دقیقه و عبارت رشته ای *"s"* تعداد ثانیه ها را بازگشت می دهد.

تابع *Datepart*

این تابع با دریافت یک داده از نوع تاریخ بخشی از آن را به صورت یک عدد صحیح که بیانگر سال ، هفته ، ماه ، روز و ... مـی باشـد بـاز مـی گرداند.
شکل کلی این تابع به صورت زیر است:

DATEPART (interval, date [, firstdayofweek [, firstweekofyear]])

این تابع دارای دو آرگومان اجباری و دو آگومان اختیاری است.

آگومان *interval* می تواند مقادیر رشته ای را که در بالا ذکر شد را بپذیرد و آرگومان *date* مقدار تاریخی است که *interval* باید از آن استخراج شود.
آرگومان های سوم و چهارم همانگونه که در مورد تابع *Datediff* توضیح داده شد عمل می کنند.

تابع *DateSerial*

این تابع سه مقدار روز ، ماه و سال را دریافت کرده و تاریخ معادل آن را باز می گرداند.

شکل کلی این تابع به صورت زیر است:

DATESERIAL (Year, Month, Day)

این تابع دارای سه آرگومان اجباری است که هر سه از نوع *integer* یا یک عبارت عددی هستند .
آگومان *year* می توانید مقادیری بین ۱۰۰ تا ۹۹۹۹ را بپذیرد.
آرگومان *year* به عنوان سال ، *Month*به عنوان ماه و *day* به عنوان روز به کار می رود.

تابع *Datevalue*

به وسیله این تابع می توان یک عبارت رشته ای را به تاریخ تبدیل نمود.
این تابع دارای یک آرگومان است که می تواند از اول ژانویه سال ۱۰۰ تا ۳۱ دسامبر ۹۹۹۹ باشد ، عبارت رشته ای می تواند حاوی یک تاریخ باشد.

شکل کلی این تابع به صورت زیر است:

DTAEVALUE (date)

در صورتی که یکی از مقادیر سال ، ماه و روز بیش از حد مجاز باشد ، هنگام اجرای برنامه پیام خطای *Type Mismatch* نمایش داده می شود.