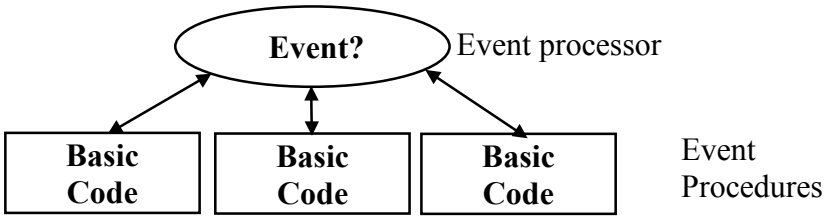
Visual Basic

۱ـ ویژوال بیسیک چیست ؟

- ویژوال بیسیک ابزاری است که برای کاربران امکان ساختن برنامه هایی برای اجرا در محیط ویندوز را فراهم می آورد. این برنامه ها به داشتن واسط کاربر گرافیکی [1]معروفند. پس از یادگیری VB به برنامه هایی مانند Word، Excel... . دیدی متفاوت خواهید داشت.
- ویژوال بیسیک برنامه ای است که بر اساس رویداد (Event) کار می کند[2]، بعبارتی کدی اجرا نمی شود، مگر آنکه رویدادی رخ دهد؛ برای مثال کاربر بر روی دکمه ای کلیک کند، منویی را انتخاب کند و غیره. ویژوال بیسیک یک پردازشگر رویداد دارد و به محض آنکه رویدادی را ردیابی کند، کد متناظر با آن رویداد را فعال می کند. که به آن " زیر برنامه رویداد " می گویند.



همه برنامه های ویندوز نیز رویداد گرا هستند. برای مثال در Word تا وقتی که کلیدی را فشار نداده اید، اتفاقی نخواهد افتاد.

---

[1] Graphic user interface (GUI)
[2] Event - driven

به دلیل همین خاصیت VB برنامه نویسی با آن ساده است. برای هر رویدادی که می خواهید، کد مربوطه را می نویسید و آن را تست می کنید.

معمولاً کدها زیاد تکرار می شوند و کافی است آنها را Copy و Paste نمایید.

- ویژگیهای دیگری از VB :

ـ دارای مجموعه کامل از کنترل هاست.

ـ تعداد زیادی تصویر و آیکون آماده دارد.

ـ به ماوس و صفحه کلید حساس است و راهنمایی های زیادی در حین برنامه نویسی می دهد.

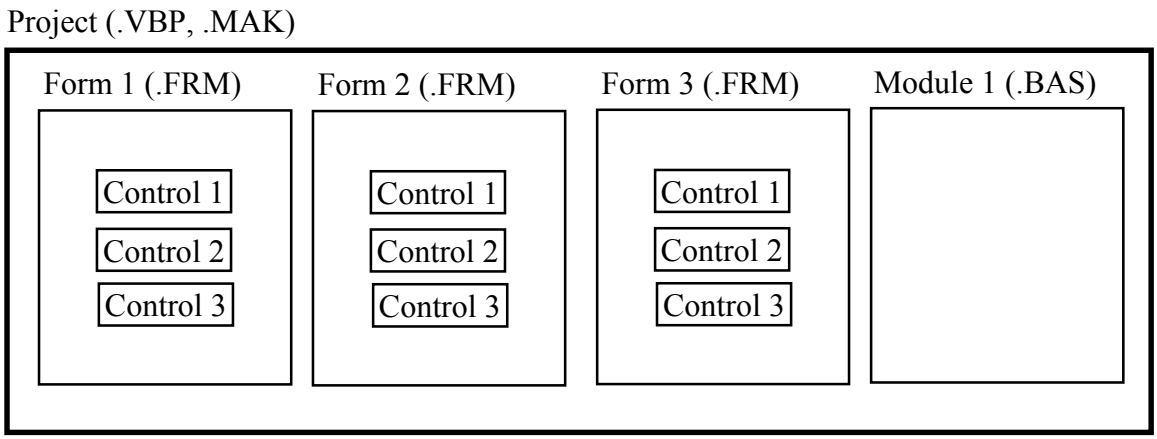
ـ جادوگری برای آماده سازی برنامه ها برای توزیع در اختیار می گذارد.

۲ـ تاریخچه VB

اولین VB در سال ۱۹۹۱ ارایه شده است. نسخه ۳ این نرم افزار دارای قابلیت های زیادی بود و در نسخه ٤، برنامه های ۳۲ بیتی پشتیبانی شده بودند. در VB نسخه ۵، امکان استفاده از کنترلهای ActiveX اضافه شده بود و برنامه های ۱۶ بیتی دیگر قابل طراحی نبودند. در نسخه ۶، امکانات قبلی پیشرفته تر شده اند و امکانات اینترنتی زیادی اضافه شده است.

برنامه هایی که با VB نسخه ۶ نوشته می شوند، در ویندوز ۹۵، ۹۸، ۲۰۰۰ و یا ویندوز NT قابل اجرا هستند.

۳ـ ساختار یک برنامه VB



هر برنامه از اجزاءِ زیر تشکیل شده است.

ـ فرمها (Forms) : پنجره هایی که می توانید برای ظاهر برنامه بسازید.

ـ کنترلها (Controls) : ویژگیهای گرافیکی که روی فرمها قرار می دهید تا کاربر با آنها کار کند؛ مانند Text box ها (جعبه های متن)، Label ها (برچسبها)، Scroll bar ها (نوارهای لغزان)، و Command button ها (دکمه های فرمان). فرمها و کنترلها، شیءِ (Object) می باشند.

ـ ویژگیها (Properties) : خصوصیات فرمها یا کنترلها را میتوان بوسیله تغییر ویژگیهای آنها تغییر داد. برای مثال آنها می توانند دارای ویژگیهایی مانند نام، عنوان، اندازه، رنگ، مکان و محتویات باشند.

ـ روشها (Methods) : زیر برنامه هایی آماده هستند که می توانند برای اجرای کاری توسط کنترلها صدا زده شوند.

زیر برنامه های رویداد (Event Procedures) : کد یا برنامه ای است که به یک شی ء مربوط می شود. این برنامه ها برای رویدادهای خاصی اجرا می شوند.

ـ زیر برنامه های عمومی (General Procedures) : این زیر برنامه ها مربوط به شی ء خاصی نیستند بلکه توسط خود برنامه صدا زده می شوند.

ـ مدولها (Modules) : مجموعه ای از زیر برنامه های عمومی، تعریف متغیرها و ثابت ها می باشند که توسط برنامه استفاده می شود.

٤ـ مراحل ایجاد برنامه

سه قدم اولیه برای ایجاد یک برنامه VB لازم است :

۱- طراحی ظاهر برنامه با قرار دادن کنترلها روی فرمها

۲- تنظیم ویژگیهای کنترلها

۳- نوشتن کد برنامه برای رویدادهای کنترلها

- با توجه به رویدادگرا بودن VB، می توانید یک زیر برنامه را بنویسید و آن را امتحان کنید تا کامل شود و سپس زیر برنامه دیگر را بنویسید. این روش خطاها را کاهش می دهد.

۵ ـ طراحی ظاهر برنامه و تنظیم ویژگیها

- VBدر سه مود کار می کند:

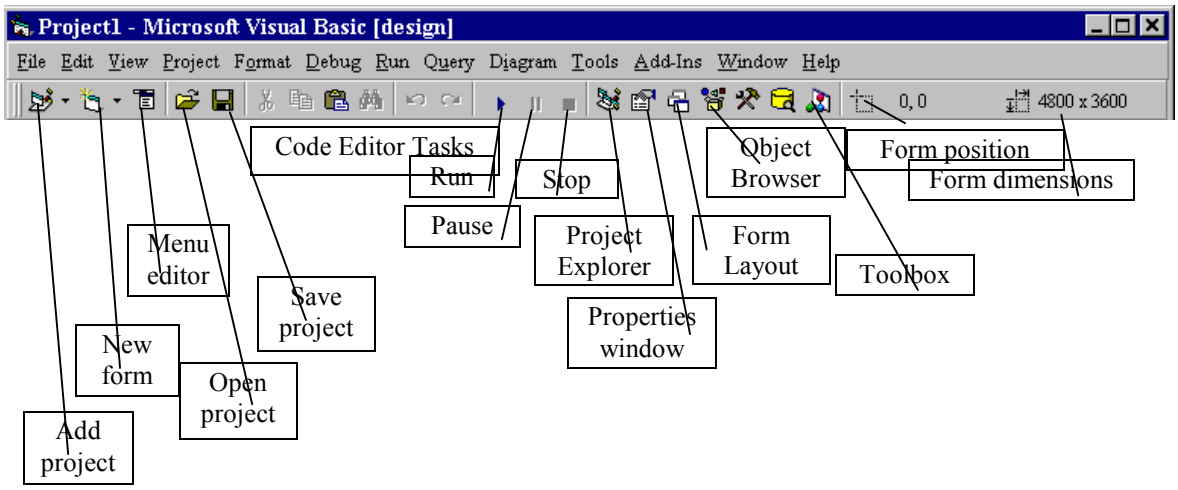
ـ مود طراحی ـ برای ساختن برنامه

ـ مود اجرا ـ برای اجرای برنامه

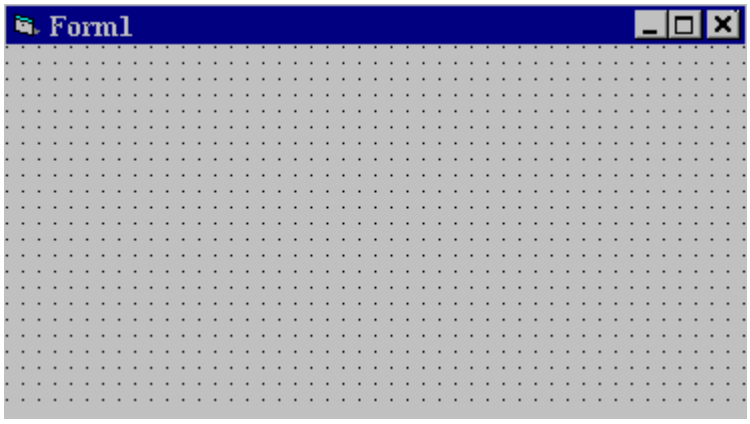
ـ مود استراحت ـ برای متوقف کردن اجرای برنامه و خطایابی آن

- با اجرای VB شش پنجره ظاهر می شوند. هر پنجره را می توان به چند روش ظاهر ساخت. انتخاب از منو، استفاده از کلیدهای تابع، یا استفاده از نوارهای ابزار. هر روشی را که با آن راحت تر هستید برگزینید.

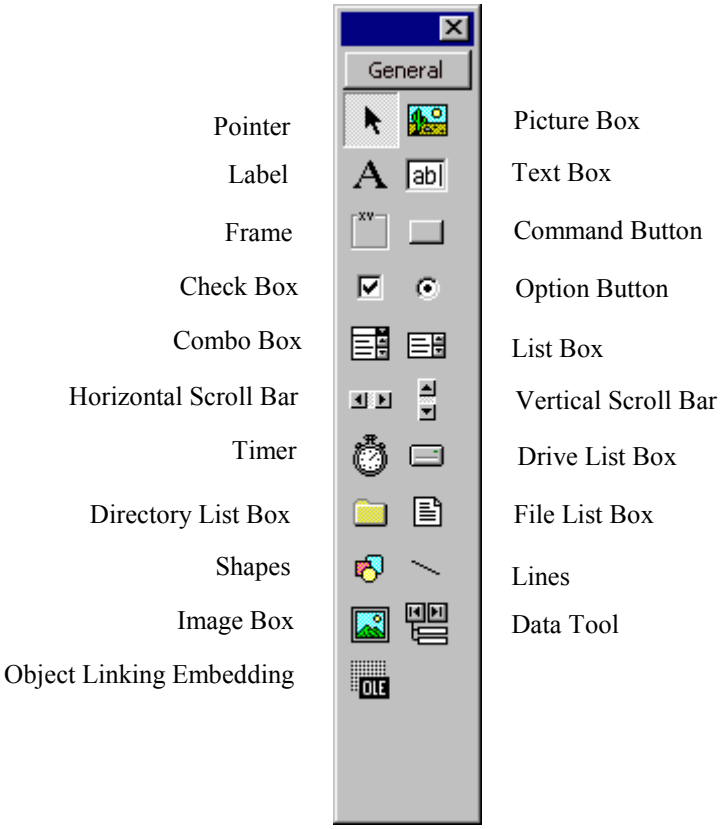
ـ پنجره اصلی (Main Window)



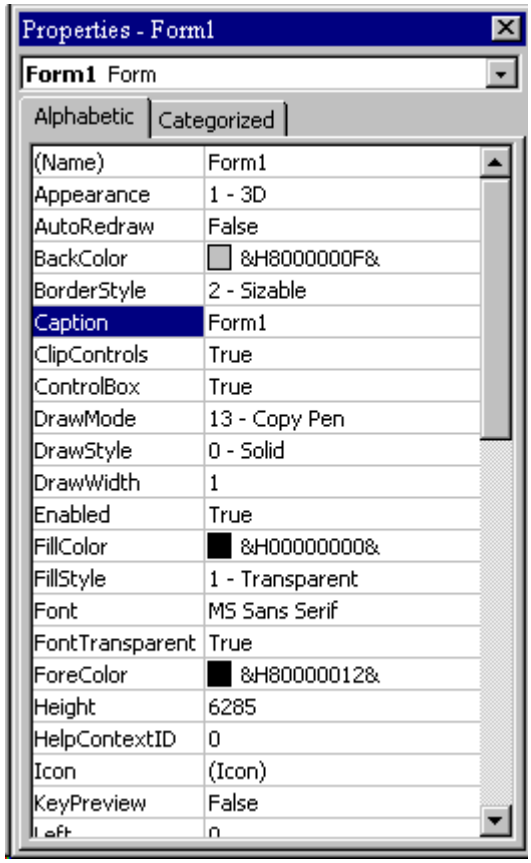
ـ پنجره فرم (Form Window)



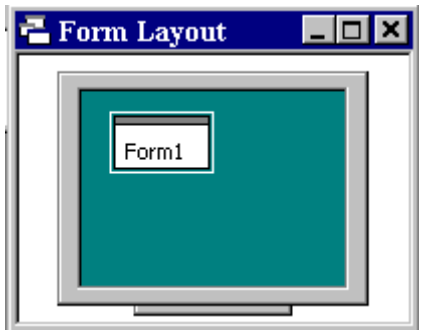
- پنجره جعبه ابزار (Toolbox)



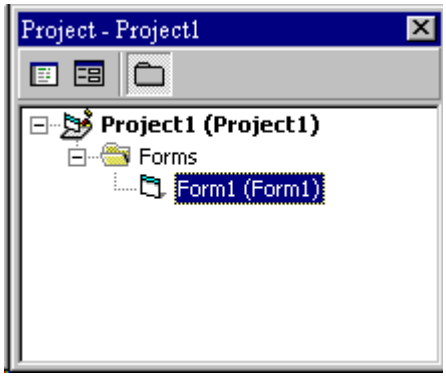
ـ پنجره ویژگیها (Properties Window)



ـ پنجره قرار گرفتن فرم (Form Layout Window)



ـ پنجره پروژه (Project Window)



- برای ایجاد ظاهر برنامه :
  ۱- کنترلها را روی فرمها قرار دهید.
  ۲- آنها را جابجا کنید.
  ۳- آنها را تغییر اندازه دهید.

۶ ـ تنظیم ویژگیهای شیءِ ها در زمان طراحی

- هر فرم یا کنترل ویژگیهایی دارد که بصورت پیش فرض به آن اختصاص داده شده اند. برای تغییر آنها در زمان طراحی، می توان از پنجره ویژگیها کمک گرفت. ویژگیها را می توان به دو صورت :

ـ مرتب شده بر اساس حروف الفبا (Alphabetic)

ـ مرتب شده بر اساس دسته (Categorized)

مشاهده کرد. یکی از ویژگیهای مهم هر کنترل Name است که بوسیله آن در برنامه به کنترل اشاره میکنیم.

- بهتر است برای نام دادن به کنترلها، از پیشوندهای معنا دار استفاده کنیم. برای مثال :

Object	Prefix	Example
Form	frm	frmWatch
Command Button	cmd, btn	cmdExit, btnStart
Label	lbl	lblStart, lblEnd
Text Box	txt	txtTime, txtName
Menu	mnu	mnuExit, mnuSave
Check box	chk	chkChoice

- نام کنترلها، می تواند ٤٠ کاراکتر طول داشته باشد. از حروف، اعداد و کاراکتر (ـ) تشکیل گردد ولی حتماً باید با یک حرف شروع شود.

۷ ـ تنظیم ویژگیها در زمان اجرا

برای تنظیم ویژگیها ی یک شی ءِ در زمان اجرا باید از شکل کلی زیر استفاده کرد :

```
ObjectName.Property = NewValue
```

برای مثال برای تنظیم ویژگی BackColor متعلق به فرم FormStart می توان نوشت :

```
FormStart.BackColor = vbBlue
```

۸ ـ استفاده از نامها برای رویداد ها کنترل

VB از نامی که به کنترلها می دهید استفاده می کند و ساختار زیر برنامه هایی را برای رویدادهای هر کنترل آماده می سازد. شکل کلی این زیر برنامه ها بصورت زیر است :

```
Private Sub ObjectName_Event (Optional Arguments)
        .
        .
End Sub
```

VB سطری را که با Sub شروع می شود، آرگومانهایش را، بعلاوه سطر End Sub آماده می کند.

۹ ـ نوشتن کد برنامه

- آخرین مرحله در ساختن برنامه نوشتن کد با زبان BASIC است. این مرحله بیشتر از هر قسمت زمان بر است. ساختار کلی زیر برنامه ها توسط VB آماده می شود. کافی است در این ساختار کد برنامه را اضافه کنیم. کد برنامه، دستوراتی است که کامپیوتر باید برای رسیدن به نتیجه ای یکی یکی اجرا نماید.
- برای نوشتن کد از پنجره کد (Code Window) استفاده می کنیم. برای دستیابی به پنجره کد می توان از منوی View، نوار ابزار یا کلید F7 استفاده کرد. از بالای پنجره کد، شیء و رویداد مورد نظر را انتخاب می کنیم تا زیر برنامه متناظر با آنها را ببینیم.

۱۰ ـ متغیرها

متغیرها برای نگهداری داده ها و اطلاعات در برنامه VB تعریف می شوند. برای نام دادن به متغیرها، قوانین زیر را باید رعایت کرد :

ـ نام متغیر باید کمتر از ٤٠ کاراکتر باشد.

ـ نام متغیر می تواند شامل حروف، اعداد و (_) باشد.

ـ نمی توان از نامهای رزرو شده از نامهایی که در VB معنای خاص دارند استفاده کرد.

۱۱ ـ انواع داده در VB

Data Type	Suffix	Example
Boolean	None	True
Integer	%	14
Long (Integer)	&	4532838
Single (Floating)	!	3.23
Double (Floating)	#	3.2346363627281
Currency	@	$12.98
Date	None	12/30/99
Object	None	n/a
String	$	“Visual Basic 6”
Variant	None	any

۱۲ ـ تعریف متغیرها

- سه روش برای تعریف متغیر ها وجود دارد :
  ۱. پیش فرض (Default)
  ۲. کوتاه (Implicit)
  ۳. مفصل (Explicit)
- اگر متغیر ها به روش کوتاه یا مفصل تعریف نشده باشند، بصورت پیش فرض نوع Variant خواهند داشت. این نوع می تواند عدد، رشته، یا داده دیگری را در خود نگه دارد.
- برای تعریف کوتاه یک متغیر، کافی است از پسوند متناظر با نوع (که در شکل فوق آمده است) استفاده نمود. برای مثال برای تعریف متغیر TextValue از نوع رشته کاراکتری :

  TextValue$= “This is a String “

  و برای تعریف متغیرAmount از نوع صحیح :

```
Amount % = 300
```

را می نویسیم.

- برای آنکه در تعریف و استفاده از نام متغیرها خطا نکنیم بهتر است از روش مفصل برای تعریف متغیرها استفاده نماییم. برای تعریف مفصل، باید دامنه (Scope) تعریف متغیر را مشخص کنیم. چهار نوع دامنه ممکن است :

ـ سطح زیر برنامه

ـ سطح زیر برنامه و استاتیک

ـ سطح فرم و مدول

ـ سطح عمومی

- برای تعریف متغیر در داخل زیر برنامه از عبارت Dim استفاده می کنیم :

```
Dim MyInt as Integer
Dim MyDouble as Double
Dim MyString as String, Yourstring as String
```

- متغیر های استاتیک، مقادیر شان را بین صدا زدن های متفاوت یک زیر برنامه حفظ می کنند. برای تعریف آنها از عبارت Static استفاده می شود.

```
Static MyInt as Integer
Static MyDouble as Double
```

- متغیرهای سطح فرم (مدول) در تمام کد فرم (مدول) و زیر برنامه های آن شناخته شده اند. آنها را در قسمت Declarations از شیءِ General فرم (مدول) در پنجره کد تعریف می کنیم و از عبارت Dim استفاده می نماییم :

```
Dim Myint as integer
Dim Mydate as date
```

- متغیرهای عمومی در تمام برنامه شناخته شده اند و مقدارشان قابل دسترسی است. برای تعریف آنها از عبارت Global استفاده می کنیم :

```
Global MyInt as Integer
Global MyDate as Date
```

- در صورتیکه دو متغیر با یک نام در دو سطح تعریف شوند متغیر محلی تر بر متغیر عمومی تر ارجحیت دارد و مقدار آن مورد استفاده قرار می گیرد.

١٤ ـ اشاره ای درباره ذخیره پروژه ها

هنگامی که پروژه VB خود را ذخیره می کنید Form ها در فایلهایی با پسوند frm، مدولها در فایلهایی با پسوند Bas و خود پروژه در فایلی با پسوند VBp ذخیره می شود. هنگام ذخیره پروژه باید دقت کنید که تمام قسمت های پروژه در محل دلخواه شما ذخیره گردند.

ساده ترین راه برای ذخیره پروژه، استفاده از دکمه Save Project روی نوار ابزار است که آیکون آن یک دیسکت فلاپی را نشان می دهد. ابتدا از شما محل ذخیره فرمها و مدولها و سپس محل ذخیره فایل پروژه سؤال خواهد شد. پس از آنکه یکبار اینکار را انجام دادید، کلیک های بعدی روی همین دکمه نوار ابزار سبب ذخیره پروژه در همان فایل خواهد شد. برای بازیابی یک پروژه ذخیره شده، می توان بر روی دکمه Open Project روی نوار ابزار (که شبیه یک پوشه فایل است) کلیک نمود.

ذخیره سازی با استفاده از دستورات منو نیز ممکن است در منوی File محیط VB چهار دستور Save بصورت زیر وجود دارد :

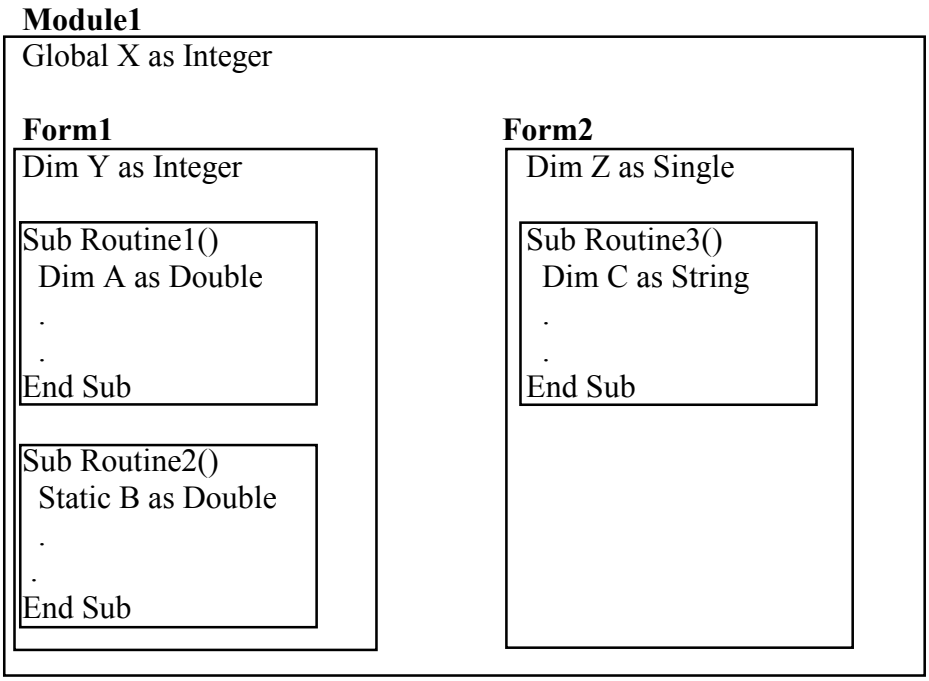
Save [Form Name]: برای ذخیره کردن فرم یا مدولی که در حال حاضر انتخاب شده است. فرم یا مدول انتخاب شده را از پنجره Project Window می توان تشخیص یا تغییر داد.

Save [Form Name] as: این منو مانند دستور قبل عمل می کند با این تفاوت که امکان تغییر نام نیز خواهیم داشت.

Save Project: این دستور همه فرمها و مدولهای برنامه را در فایلهای متناظر شان و همچنین خود پروژه را در فایل پروژه ذخیره می کند.

Save Project As: این گزینه امکان ذخیره سازی فرمها، مدولها و فایل پروژه و همچنین تغییر نام فایل آنها را فراهم می سازد.

همچنین می توان از گزینه Open در منوی File برای بازیابی یک پروژه ذخیره شده استفاده کرد.



۱۵ ـ تاریخچه برنامه نویسی به زبان بیسیک

نوشتن کد در محیط VB به زبان بیسیک انجام می شود. نام بیسیک از کلمات زیر گرفته شده است:

Beginner's All–Purpose Symbolic InstrutionC_ode

این زبان در اواسط دهه هفتاد توسط دو دانشجو نوشته شد. این دو دانشجو بیل گیتز [1] و پل الن [2] بودند.

بعدها نسخه های دیگری از این زبان مانند GW-Basic، QBasic، Quick Basic و سرانجام Visual Basic بر اساس بیسیک اولیه نوشته شدند.

[1] Bill Gates
[2] Paul Allen

۱۶ ـ عبارات VB

- ساده ترین عبارت VB، عبارت مقدار دهی (Assignment Statement) است. در این عبارت به یک متغیر مقداری داده می شود. برای این کار نام متغیر نوشته شده، پس از آن علامت = و مقدار جدید نوشته می شود. مقدار جدید می تواند خود یک عبارت ترکیبی باشد. برای مثال :

```
StartTime = Now
Explorer.Caption = "Caption Spaulding "
BitCount = ByteCount * 8
Energy = Mass * LIGHTSPEED ^2
NetWorth = Assets – Liabilities
```

- هر عبارت بطور معمول روی یک خط نوشته می شود و نیازی به علامت اتمام عبارت نیست.

اگر بخواهیم چند عبارت را روی یک خط بنویسیم از علامت (:) برای جدا کردن آنها استفاده می کنیم. برای مثال :

```
StartTime = Now: EndTime = StartTime +10
```

به این کار اصطلاحاً Stack کردن عبارات می گویند. این کار همیشه مطلوب نیست و در عبارت If/End If که خواهیم گفت می تواند خطا ایجاد کند.

- ممکن است عبارتی طولانی تر از یک خط باشد. برای آنکه ادامه آن را در خط بعد بنویسیم از (_) در انتهای خط استفاده میکنیم. به کاراکتر Underscore در این موارد کاراکتر ادامه (Continuation Character) نیز گفته می شود.

برای مثال :

```
Months = Log (Final * IntRate / Deposit +1)_
        / Log (1+ IntRate)
```

- یکی از تکنیک های خوب برنامه نویسی نوشتن توضیحات در کد برنامه است. اینکار باعث می شود که خطایابی برنامه و یا تغییر آن توسط خود برنامه نویس یا برنامه نویس دیگر ساده شود زیرا به فهم برنامه کمک می کند. برای نوشتن توضیحات، بوسیله کلمه Rem یا علامت (‘) به VB می گوییم که در ادامه توضیح خواهیم داد تا VB سعی در اجرای نوشته های ما نکند. برای مثال :

```
Rem This is a remark
Rem This is also a remark
 X=2*y 'another way to write a remark or comment
```

۱۷ـ عملگرهای VB

- ساده ترین عملگرها برای محاسبات ریاضی بکار می روند. این عملگرها به ترتیب اولویت عبارتند از :

عملگر	عمل انجام شده
^	توان
* /	ضرب و تقسیم
/	تقسیم صحیح (قسمت اعشاری حذف می شود)
Mod	باقی مانده تقسیم
+ -	جمع و تفریق

- از پرانتزها میتوان برای تغییر ترتیب اولویت استفاده کرد.

- عملگر & یا + برای اتصال دو رشته کاراکتری به هم بکار می روند. برای مثال :

```
LblTime.Caption = " The current time is " & Format (Now, " hh: mm ")
txtSample.Text = " Hook This " + " To This "
```

- شش عملگر مقایسه ای نیز در VB وجود دارد :

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

- نتیجه هر مقایسه یک عبارت بولین درست (True) یا نادرست(False) است.
- از سه عملگر منطقی زیر می توان در VB استفاده کرد.

عملگر	عمل
Not	not منطقی
And	and منطقی
Or	or منطقی

- عملگر Not یک عملوند را برعکس می کند. بعبارتی True را False و False را True می کند.
- عملگر And در صورتی که هر دو عملوندش True باشند، مقدار True را بر می گرداند و در غیر این صورت پاسخ False می دهد.
- عملگر Or در صورتی که حداقل یکی از عملوند هایش True باشد، مقدار True بر می گرداند و اگر هر دو عملوندش False باشند، پاسخ False می دهد.
- عملگرهای منطقی بر عملگرهای حسابی اولویت دارند.

۱۸ـ توابع VB

توابع زیر در VB موجودند :

تابع	مقداری را که تابع بر می گرداند
Abs	قدر مطلق یک عدد
Asc	کد ASCII یک کاراکتر
Chr	کاراکتر معادل یک کد اسکی
Cos	سینوس یک زاویه
Format	تاریخ یا عدد که بصورت یک رشته کاراکتری آمده است.
InStr	مکان یک حرف یا رشته در یک رشته دیگر
Left	قسمت انتخابی از چپ یک رشته
Len	تعداد کاراکترهای یک رشته
Mid	قسمت انتخابی از یک رشته
Now	زمان و تاریخ فعلی سیستم

Right	قسمت انتخابی از راست یک رشته
Rnd	یک عدد تصادفی
Sin	سینوس یک زاویه
Sqr	جذر یک عدد
Str	رشته کاراکتری معادل یک عدد
Timer	تعداد ثانیه هایی که از نیمه شب گذشته است.
Trim	فاصله های خالی چپ و راست یک رشته را حذف کرده، رشته باقیمانده را بر می گرداند.
Val	ارزش عددی یک رشته کاراکتری

۱۹- توابع رشته ای

- در VB توابع زیادی برای کار کردن با رشته های کاراکتری وجود دارد. ویژگی Caption کنترل Label و ویژگی Text کنترل Textbox از نوع رشته های کاراکتری هستند. می توان توسط کنترل Label یک رشته کاراکتری را به کاربر نشان داد، و یا توسط کنترل Textbox یک رشته کاراکتری را از کاربر پرسید.

- برای تبدیل یک رشته کاراکتری به یک مقدار عددی می توان از تابع Val استفاده کرد. بعنوان مثال برای پرسیدن عددی که کاربر در Textbox با نام TxtExample وارد می کند، و قرار دادن آن در متغیر X می توان از دستور زیر استفاده کرد :

```
X=Val ( txtExample.Text )
```

- برای تبدیل یک عدد به یک رشته کاراکتری دو راه وجود دارد. راه اول استفاده از تابع Str است که تبدیل را بدون توجه به نحوه نشان دادن عدد به کاربر انجام می دهد. برای مثال اگر بخواهیم عدد MyNumber را در Textbox ای با نام TxtExample به کاربر نشان دهیم :

```
MyNumber = 3.14159
TxtExample.Text = Str ( MyNumber )
```

از طرف دیگر اگر بخواهیم تعداد اعداد اعشاری یا سایر موارد را در هنگام نشان دادن عدد کنترل نماییم، می توانیم از تابع Format استفاده کنیم. این تابع دارای دو آرگومان است. آرگومان اول، عددی هست که می خواهیم تبدیل کنیم. آرگومان دوم، یک رشته است که نحوه تبدیل عدد را مشخص می کند. برای مثال برای نشان دادن عدد MyNumber با تنها دو عدد اعشاری، بصورت زیر عمل می کنیم :

```
MyNnumber = 3.14159
TxtExample.text = format ( MyNumber، “ #. # # “)
```

- می توان به کمک تابع Left از سمت چپ هر رشته به تعداد دلخواه حروف را جدا کرد. برای مثال برای جدا کردن سه حرف از رشته Mystring و قرار دادن آن سه حرف در متغیر Leftstring بصورت زیر عمل می کنیم :

```
Mystring = “ visval basic is fun! “
Leftstring = left ( MyString، 3 )
```

در مثال فوق متغیر Leftstring حاوی رشته " Vis " خواهد شد.

- تابع Right عکس تابع Left عمل می کند، بعبارتی حروف را از سمت راست رشته جدا می کند. در مثال زیر ۶ حرف از سمت راست رشته Mystring جدا کرده ایم :

Rightstring = right(MyString,6)

این دستور رشته " s fun! " را در متغیر Rightstring قرار می دهد.

- تابع Mid امکان جداسازی حروف از هر جای رشته را فراهم می سازد. سه آرگومان این تابع عبارتند از : رشته مورد نظر، محل شروع جداسازی حروف، و تعداد حروفی که باید جدا شود.

برای مثال :

MidString = mid (mystring, 3,6)

این دستور از حرف سوم شروع کرده، شش حرف را در متغیر Midstring قرار می دهد، بعبارتی متغیر Midstring حاوی رشته " sual B " خواهد بود.

- تابع Len حروف موجود در یک رشته را می شمرد :

Lenstring = len(mystring)

با توجه به رشته Mystring، Lenstring برابر ۲۰ خواهد شد.

- برای یافتن یک رشته در یک رشته دیگر می توان از تابع Instr استفاده کرد. این تابع سه آرگومان دارد : مکان شروع جستجو در رشته اول (که آرگومانی اختیاری است)، رشته اول (که جستجو در آن صورت میگیرد)، رشته دوم (که به دنبال آن هستیم). تابع مکان اولین حرف از رشته دوم را در رشته اول اعلام میکند. اگر این رشته را نیابد، مقدار صفر برگردانده می شود. برای مثال :

Location = insert (3, mystring, “ sic “)

این دستور به این معناست که می خواهیم در رشته Mystring رشته " sic " را پیدا کنیم. جستجو را از حرف سوم شروع می کنیم. اگر محل شروع را مشخص نکنیم، جستجو از حرف اول انجام خواهد گرفت. با توجه به محل sic در رشته، Location برابر عدد ۱۰ خواهد شد.

- دو تابع مفید دیگر، توابع Asc، Chr می باشند. می دانیم که هر کاراکتر در کامپیوتر با یک عدد نشان داده می شود. برای اینکه اطلاعات کامپیوترها قابل انتقال به هم باشد، از استانداردی به نام استاندارد ASCII (American Standard Code for Information Interchange)

برای ذخیره حروف استفاده می شود. تابع Asc عدد متناظر با هر حرف را بر میگرداند. برای مثال

Asc (“A”)

کد اسکی حرف A (که ۶۵ است) را اعلام می کند. تابع Chr، برعکس، کد را گرفته، کاراکتر متناظر با آن را برمی گرداند. برای مثال:

Chr (48)

حرف معادل کد ٤٨ اسکی (که حرف " 1 " است) را اعلام می کند.

- توابع Lcase، Ucase برای تبدیل حروف رشته به حروف کوچک یا بزرگ استفاده می شوند. برای مثال

a = “ aBcdE12 “
b = lcase (a)
c = ucase (a)

این دستورات باعث می شوند که b حاوی رشته" abcde12" و c حاوی رشته" ABCDE12 "شود.

- توابع Rtrim و Ltrim فواصل خالی سمت راست و چپ یک رشته را حذف می کنند. برای مثال :

A = “ abc “
B = “ * “ + R trim (a) + “ * “
C = “ * “ + L trim (a) + “ * “

این دستورات سبب می شوند که b برابر " *abc " * و c بر ابر "* abc * " شود.

- تابع String، به تعداد مورد نظر از یک کاراکتر تکرار می نماید. آرگومان اول این تابع تعداد کاراکتر و آرگومان دوم کاراکتر مورد نظر یا کد اسکی آن می باشد. برای مثال هر یک از دو دستور زیر ۱۰ حرف A در رشته a قرار می دهد.

a = string (10, “A“)
a = string (10, 65)

- تابع Space، به تعداد مورد نظر فاصله خالی تکرار می کند. برای مثال:

a = “ * “ + Space (10) + “ * “

این دستور رشته "* * " را در a قرار خواهد داد.

۲۰- ایجاد اعداد تصادفی

- برای نوشتن بازیها یا تست کردن بعضی از برنامه ها لازم است بتوانیم اعداد تصادفی ایجاد کنیم. برای مثال هنگام انداختن تاس انتظار یک عدد تصادفی بین یک یا شش را داریم. در VB تابع Rnd یک عدد تصادفی بین 0 تا 1 ایجاد می کند. برای اینکه یک عدد صحیح بین Imin و Imax داشته باشیم، می توانیم از رابطه زیر استفاده کنیم :

I = Int ((Imax – Imin + 1) * Rnd) + Imin

(تابع Int قسمت صحیح اعداد اعشاری را جدا می کند).

- برای آنکه اعداد تصادفی توسط تابع Rnd ایجاد شوند، Vb از یک هسته برای تولید اعداد استفاده می کند. این هسته را می توان بکمک تابع Randomize تعیین کرد :

Randomize seed

برای آنکه اعداد کاملاً تصادفی باشند و تغییر کنند، هسته نیز باید متغیر باشد. می توان از ساعت سیستم بعنوان هسته ایجاد اعداد تصادفی استفاده کرد :

Randomize timer

می توان Timer را هم در عبارت فوق حذف کرد، زیرا پیش فرض VB است :

Randomize

- **مثال :**

برای شبیه سازی کردن انداختن یک تاس :

NumberSpots = Int (6 * Rnd) + 1

برای انتخاب یک عدد بین ۱۰۰ تا ۲۰۰ :

Number = Int (101* Rnd) + 100

۲۱- ثابتها در VB

ثابت بر خلاف متغیر، دارای مقداری مشخص و غیر قابل تغییر است. ثابتها را می توان تعریف نمود و یا از ثابتها تعریف شده در VB استفاده کرد.

الف) ثابتهای تعریف شده یا سمبولیک (Symbolic Constants)

- برای آنکه ثابتهایی که در برنامه ها استفاده می شوند، معنادار تر باشند، وخواندن برنامه هایی که از آنها استفاده می کنند آسانتر باشد، تعدادی از ثابتها در VB تعریف شده اند و نام خاص دارند.
- برای مثال، برای آنکه زمینه یک فرم با نام FormExample را به رنگ آبی در آوریم، کافیست دستور زیر را بنویسیم :

```
FormExample. BackColor = OxFF0000
```

و یا آنکه به جای استفاده از عدد معادل رنگ آبی در دستور فوق، می توانیم از ثابت vbBlue استفاده کنیم:

```
FormExample. BackColor = vbBlue
```

استفاده از ثابتهای VB، هر جا که ممکن باشد توصیه می شود.

ب) تعریف ثابتهای دلخواه

- می توان ثابتهای دلخواه را نیز در VB تعریف کرد. برای مثال برای تعریف عدد PI معادل 3.14159 دستور زیر را می نویسیم :

```
Const PI = 3.14159
```

- تمام نام ثابتها، باید از حروف بزرگ تشکیل شده باشد تا بتوان آنها را از متغیر تمیز داد. دامنه ثابتها نیز مانند دامنه متغیرها تعریف می شود. برای مثال، اگر ثابتی را داخل یک زیر برنامه (Procedure) تعریف کنید، فقط در همان زیر برنامه شناخته شده است. اگر می خواهید ثابتی را تعریف کنید که در کل برنامه شناخته شده باشد، از دستور زیر استفاده نمایید :

```
Global Const PI = 3.14159
```

دستور فوق در قسمت General Declaration یک مدول باید نوشته شود.

۲۲ ـ دستور IF

- برخی مواقع در برنامه ها لازم است در صورت برقرار بودن شرط خاصی، دستوری اجرا شود. در این مواقع ساده ترین دستور، دستورIF/ Then در VB است، برای مثال :

```
IF Balance - check < 0 Then Print " You are overdrawn"
```

در دستور فوق اگر وفقط اگرBalance - check از صفر کوچکتر باشد، جمله" You are Withdrawn " نوشته خواهد شد.

- در صورتیکه بخواهیم در صورت برقرار بودن شرطی، چند دستور اجرا شود، می توانیم از قطعه برنامه هایی به شکل IF / Then / End IF استفاده نماییم. برای مثال :

```
IF Balance - check <0 then
        Print " You are overdrawn"
        Print " Authorities Have Boon Notified."
```

```
End IF
```

در مثال فوق در صورتیکه Balance - check از صفر کوچکتر باشد، هر دو سطر پرینت خواهند شد.

- قطعه برنامه هایی به شکل IF/ Then / Else / End IF نیز ممکنند :

```
IF Balance - Check <0 Then
    Print " Oops "
    Print " Oh Oh "
Else
    Balance = Balance - Check
End If
```

در مثال فوق، در صورتیکه Balance - Check از صفر کوچکتر باشد، دو سطر قید شده پرینت خواهد شد، در غیر این صورت (اگر کوچکتر از صفر نباشد) مقدار جدید Balance محاسبه خواهد شد.

- می توان از عبارت Else IF نیز استفاده کرد، برای مثال :

```
IF Balance - Check <0 Then
    Print " oops "
    Print " Oh Oh "
Else If Balance - Check = 0 Then
    Print " Whew! You Barely Made It "
    Balance = 0
Else
    Balance = Balance - Check
End If
```

در مثال فوق، علاوه بر دستورات برنامه قبل، در دستوریکه Balance - Check برابر صفر باشد، پیغام " Whew.. " پرینت خواهد شد.

- در نوشتن هر دستور IF، توجه نمایید که همه حالات ممکن را در نظر گرفته باشید. بعلاوه دقت کنید که اولین باری که دستورIF شرط صحیحی پیدا کند، دستورات مربوطه را اجرا کرده، ازIF خارج می شود. در نتیجه اگر شرطهای دیگری نیز صحیح باشند، دستورات آنها اجرا نخواهد شد.

۲۳ـ به دام اندازی کلیدهای کیبرد

- اگر بخواهیم درText Box ها کاربر کاراکترهای خاصی را وارد کند برای مثال اعداد را بنویسد و حروف را وارد نکند، لازم است کلیدهایی را که روی کی برد فشار می دهد کنترل کنیم. فرایند کنترل و ممانعت از کلیدهای ناخواسته را اصطلاحاً به راه اندازی کلیدهای کی برد و یا (Key Trapping) می نامیم.

- دستورات لازم را باید در زیر برنامه KeyPress کنترل مربوطه وارد کنیم. برای مثال برای textbox ای بنام TxtText، در زیر برنامه :

```
Private Sub TxtText_ KeyPress (KeyASCII As Integer)
        ▪
        ▪
        ▪
 End Sub.
```

هر بار کلیدی روی صفحه کی برد فشرده می شود، کد اسکی (ASCII Code) مربوط به کلید فشرده شده به این زیر برنامه فرستاده می شود (KeyASCII). اگر KeyASCII مربوط به یک کلید قابل قبول باشد، دستوری لازم نیست انجام شود. در غیر این صورت KeyASCII را برابر صفر قرار می دهیم و از زیر برنامه خارج می شویم. اینکار باعث می شود که فشردن کلید نادیده گرفته شود. مقادیر ASCII همه کلیدها در VB Help موجودند. بعلاوه برخی ثابتهای سمبولیک معادل نیز دارند.

- بعنوان مثال فرض کنید که یک Text Box بنام TxtExample داریم و می خواهیم فقط حروف بزرگ از A تا Z در آن نوشته شوند. کد ASCII معادل این حروف ۶۵ تا ۹۰ می باشد، ولی ثابتهای سمبولیک vbKeyA و vbKeyZ نیز معادل همین اعداد هستند. زیر برنامه را می توانیم بصورت زیر کامل کنیم (اگر کلیدی اشتباه وارد شود، صدای بیپ شنیده خواهد شد) :

```
Private Sub txtExample_KeyPress(KeyAscii as Integer)
If KeyAscii >= vbKeyA And KeyAscii <= vbKeyZ Then
  Exit Sub
Else
  KeyAscii = 0
  Beep
End If
End Sub
```

- بهتر است در زیر برنامه فوق، به کاربر اجازه دهیم از کلید Backspace نیز برای پاک کردن و اصلاح آنچه تایپ کرده استفاده نماید. کد اسکی این کلید ۸ است و ثابت سمبولیک vbKeyBack نیز معادل همین عدد است .

۲۴- دستور SelectCase

- علاوه بر دستورIF، می توان از دستور Select Case نیز برای هدفی مشابه استفاده نمود.
- فرض کنید دستورات زیر را با دستور IF نوشته ایم :

```
If Age = 5 Then
  Category = "Five Year Old"
ElseIf Age >= 13 and Age <= 19 Then
  Category = "Teenager"
ElseIf (Age ≥ 20 and Age <= 35) Or Age = 50 Or (Age >=
60 and Age <= 65) Then
  Category = "Special Adult"
ElseIf Age > 65 Then
  Category = "Senior Citizen"
Else
  Category = "Everyone Else"
End If
```

معادل این دستورات، با Select Case بصورت زیر است :

```
Select Case Age
Case 5
  Category = "Five Year Old"
Case 13 To 19
  Category = "Teenager"
Case 20 To 35, 50, 60 To 65
  Category = "Special Adult"
Case Is > 65
  Category = "Senior Citizen"
Case Else
  Category = "Everyone Else"
End Select
```

۲۵- دستور GoTo

- یک راه دیگر برای جابجا شدن و شاخه به شاخه کردن دستوارات در VB دستور GoTo است. این دستور، یکی از منفورترین دستورات نزد برنامه نویسان است. با اینهمه برای کنترل مواقعی که برنامه خطا دارد، از این دستور استفاده می شود.
- در صورتیکه دستور GoTo Label را داشته باشیم، اجرای برنامه پس از این دستور به سطری از برنامه منتقل خواهد شد که دارای برچسب Label است. برچسب هر سطر در ابتدای آن سطر نوشته شده، بکمک (:) از دستور آن سطر جدا می شود.
- برای مثال :

```
Line10:
    .
    .
GoTo Line10
```

هر گاه به دستورGoTo برسیم، اجرای برنامه دوباره به سطری که برچسب Line l0 دارد منتقل خواهد شد.

۲۶- حلقه ها در VB

- حلقه ها برای اجرای دستوراتی نوشته می شوند که باید چندین بار تکرار شوند. حلقه با برقراری شرطی در ابتدا یا انتهای حلقه اتمام می یابد.

- حلقه بصورت Do / While Loop :

```
Counter = 1
Do While Counter <= 1000
  Debug.Print Counter
  Counter = Counter + 1
Loop
```

این حلقه تا وقتی که (While) متغیر Counter کوچکتر یا مساوی ۱۰۰۰ است تکرار می شود. دقت کنید که حلقه Do While / Loop در صورت عدم برقراری شرط While حتی یکبار هم اجرا نخواهد شد. دستور Debug .Print مقدار Counter را در پنجره Debug برنامه VB نشان می دهد.

- حلقه بصورت Do Until / Loop :

```
Counter = 1
Do Until Counter >  1000
  Debug.Print Counter
  Counter = Counter + 1
Loop
```

این حلقه تکرار می شود تا (Until) متغیر Counter از ۱۰۰۰ بیشتر شود. (دقت کنید که اگر هنگام شروع شرط Until برقرار باشد، حلقه حتی یکبار نیز اجرا نخواهد شد.

- حلقه بصورت Do / Loop While:

```
Sum = 1
Do
  Debug.Print Sum
  Sum = Sum + 3
Loop While Sum <= 50
```

این حلقه تا وقتی که (while) متغیر Sum کوچکتر یا مساوی ۵۰ است، ادامه می یابد. توجه کنید که از آنجاییکه بررسی حلقه در انتهای حلقه انجام میشود، حلقه حداقل یکبار انجام خواهد شد.

- حلقه بصورت Do / Loop Until:

```
Sum = 1
Do
  Debug.Print Sum
  Sum = Sum + 3
Loop Until Sum > 50
```

این حلقه ادامه می یابد تا (Until) متغیر Sum بزرگتر از ۵۰ شود. در این مثال نیز، حلقه Do/ Loop Until همیشه حداقل یکبار تکرار خواهد شد.

- توجه کنید که حلقه ها باید حتماً به نحوی تمام شوند. در صورتیکه حلقه ای بی انتها تکرار شود، کامپیوتر حالت Hang پیدا می کند. اگر برای شما چنین مشکلی آمد، Ctrl + Break را امتحان کنید. اگر فایده ای نداشت ناچارید کامپیوتر خود را Reboot کنید.

- دستور Exit Do باعث خروج از حلقه می شود و اجرای برنامه از دستور بعد از حلقه ادامه پیدا می کند.

۲۷ـ شمارش در VB

- توسط حلقه For/ Next می توان شمارش انجام داد. برای مثال :

```
For I = 1 to 50 Step 2
  A = I * 2
  Debug.Print A
```

```
Next I
```

در ابتدای حلقه فوق، I مقدار یک دارد. هر بار که حلقه می خواهد تکرار شود، I بعلاوه ۲ می شود (زیرا Step 2 قید شده است). این حلقه تکرار می شود تا I برابر مقدار نهایی خود (که ۵۰ است) شود. اگر عبارت Step ذکر نشود، I هر بار یکی بیشتر می شود. بعلاوه Step های منفی هم مجاز هستند.

- می توان از یک حلقه For/ Next با دستور Exit For خارج شد. اجرای برنامه از دستور بعد از Next ادامه پیدا خواهد کرد.

۲۸ـ جعبه پیام (Message Box)

- یکی از بهترین توابع در VB، Message Box است. این دستور یک پیام، یک آیکون و تعدادی دکمه فرمان را نشان می دهد. کاربر می تواند با زدن یک دکمه پاسخ گوید.

- شکل عبارتی این دستور، مقداری بر نمی گرداند و فقط پیغام را نشان می دهد :

  Message Box Message, Type, Title

  که در آن :

  Massage: پیامی را که می خواهیم نشان دهیم.

  Type: نوع جعبه پیام.

  Title: متنی که نوار عنوان جعبه پیام نشان داده می شود.

  نمی توان محل جعبه پیام را روی صفحه تعیین نمود.

- شکل تابعی این دستور، یک مقدار صحیح را بر می گرداند که متناظر با کلیدی است که کاربر کلیک کرده است. برای مثال :

```
Dim Response As Integer
Response = MsgBox (Message، Type، Title)
```

- پارامتر Type مجموع چهار مقدار است که متناظرند با دکمه هایی که باید نشان داده شوند، آیکونی که باید نمایش داده شود، دکمه ای که جواب پیش فرض است، و مودالیتی (Modality) جعبه پیام.

- اولین جزءِ از مقدار Type، دکمه های فرمان را تعیین می کند :

Value	Meaning	Symbolic Constant
0	OK button only	vbOKOnly
1	OK/Cancel buttons	vbOKCancel
2	Abort/Retry/Ignore buttons	vbAbortRetryIgnore
3	Yes/No/Cancel buttons	vbYesNoCancel
4	Yes/No buttons	vbYesNo
5	Retry/Cancel buttons	vbRetryCancel

- دومین جزءِ از مقدار Type، آیکونی که باید نمایش داده شود را مشخص می نماید:

Value	Meaning	Symbolic Constant
0	No icon	(None)
16	Critical icon	vbCritical

32	Question mark	vbQuestion
48	Exclamation point	vbExclamation
64	Information icon	vbInformation

- سومین جزءِ مشخص می کند که کدام دکمه از دکمه های فرمان پیش فرض می باشد، بعبارتی فشردن کلید Enter معادل کلیک کردن روی دکمه پیش فرض است.

Value	**Meaning**	**Symbolic Constant**
0	First button default	vbDefaultButton1
256	Second button default	vbDefaultButton2
512	Third button default	vbDefaultButton3

- آخرین جزءِ، مودالیتی جعبه پیام را مشخص می کند .

Value	**Meaning**	**Symbolic Constant**
0	Application modal	vbApplicationModal
4096	System modal	vbSystemModal

اگر جعبه، Application Modal باشد، کاربر باید به جعبه پیام پاسخ گوید تا بتواند برنامه اش را ادامه دهد و بعبارتی ادامه برنامه، فقط در صورت پاسخگویی به جعبه پیام ممکن است. اگر جعبه System Modal باشد، همه برنامه ها منتظر جواب به جعبه پیام خواهند ماند.

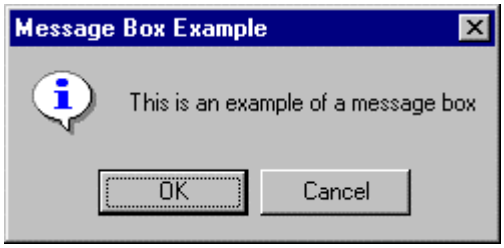
- با توجه به آنکه مقادیر ممکن برای Type دارای ثابت های سمبولیک نیز هستند. بهتر است از این ثابت ها استفاده شود.

- مقداری که توسط جعبه پیام برگردانده می شود، مشخص کننده دکمه ای است که کاربر کلیک نموده است. این مقادیر بصورت زیر می باشند :

Value	**Meaning**	**Symbolic Constant**
1	OK button selected	vbOK
2	Cancel button selected	vbCancel
3	Abort button selected	vbAbort
4	Retry button selected	vbRetry
5	Ignore button selected	vbIgnore
6	Yes button selected	vbYes
7	No button selected	vbNo

مثال :

```
MsgBox "This is an example of a message box", vbOKCancel + vbInformation,
                                                "Message Box Example"
```



۲۹ـ روشها یا متدهای شیء (Object Methods)

- می دانیم که هر کنترل یا شیء دارای ویژگیها (Properties) و رویدادهایی (Events) است که در ارتباط با آن وجود دارند. علاوه بر ایندو هر شیء دارای تعدادی روش یا متد (Method) است. یک متد، زیر برنامه ای یا تابعی است که عملی را روی شی ء انجام می دهد.

- در ادامه میخواهیم هر کنترل و متدهای آن را بررسی کنیم. متدها در زمان اجرا بکار می آیند. نحوه صدازدن یک متد بصورت زیر است :

{پارامترهای انتخابی} متد. نام شی ء
ObjectName.Method {Optional Arguments}

۳۰ـ شیء Form

- هر Form ناحیه ای است که کاربر با آن سر و کار دارد و یکی از اصلی ترین اجزاء در VB است.
- ویژگی های Form :

Appearance: برای انتخاب ظاهر تخت یا سه بعدی.
BackColor: برای انتخاب رنگ زمینه.
BorderStyle: برای انتخاب چگونگی مرز Form که آیا ثابت باشد یا بتوان اندازه آن را تغییر داد.
Caption: عنوان فرم که در نوار عنوان پنجره دیده می شود.
Enabled: اگر مقدار آن True باشد، به کلیک های ماوس و دکمه های صفحه کلید پاسخ میدهد، در صورتیکه False باشد، غیر فعال می شود.
Font: برای تنظیم قلم، اندازه و نوع آن.
ForeColor: برای تنظیم رنگ متن یا تصاویر.
Picture: بکمک این ویژگی میتوان یک تصویر (از نوع Bitmap) را روی فرم قرار داد.
Visible: اگر مقدار آن False باشد، فرم مخفی میگردد.

- رویدادهای Form:

Activate: رویداد Form_Activate زمانی صدا زده میشود که پنجره فرم، فعال گردد.
Click: رویداد Form_Click زمانی اجرا میشود که کاربر روی فرم کلیک کرده باشد.
DblClick: رویداد Form_DblClick زمانی اجرا میشود که کاربر روی فرم دابل کلیک کرده باشد.

Load: رویداد Form_Load زمانی صدا زده میشود که فرم در حافظه بار شود. این زیربرنامه بهترین جا برای مقدار اولیه دادن به پارامترهای فرم است.

- **متدهای فرم :**

Cls: این متد کلیه متن ها و گرافیک را از روی فرم پاک می کند. البته هیچ شیئی با این متد حذف نمی گردد.

Print: با این متد می توان یک رشته متن را روی فرم چاپ نمود.

مثال : این دستور فرم را پاک می کند.

```
FrmExample.Cls
FrmExample.Print  "This Will Print On The Form "
```

این دستور سبب چاپ جمله This Will Print On The Form روی فرم FrmExample می گردد.

- **شبکه های نقطه چین (Grid Lines)**

بر روی فرم، نقاط ریزی بصورت یک شبکه نقطه چین قرار گرفته اند. این نقاط به طراح کمک میکنند که اشکال و عناصر را بصورت مناسب و هماهنگ در فرم قرار دهد. زیرا هنگام حرکت و جابجایی یک عنصر، نقطه شروع (گوشه بالاـ چپ عنصر) فقط در رأس این نقاط خواهد بود. این نقاط فقط در زمان طراحی وجود داشته، در هنگام اجرای برنامه ناپدید می شوند.

برای تنظیم این شبکه از منوی Tools گزینه Options را انتخاب کرده، به صفحه General بروید. چهار گزینه مربوط به تنظیم شبکه نقاط عبارتند از :

- Grid Width (عرض شبکه نقطه چین): فاصله افقی نقاط از همدیگر بر حسب Twip. مقدار پیش فرض ۱۲۰ می باشد. (در VB از واحد Twip به مفهوم یک بیستم نقطه یا Twentieth Of a Point جهت اندازه گیری استفاده می شود. هر Twip معادل $\frac{1}{1440}$ اینچ می باشد. در صنعت چاپ هر نقطه Point معادل $\frac{1}{72}$ اینچ است .)
- Grid Height (ارتفاع شبکه) : فاصله عمودی نقاط از همدیگر بر حسب Twip. فاصله پیش فرض ۱۲۰ می باشد.
- Show Grid (نمایش شبکه) : به کمک این گزینه می توان نقاط شبکه را در زمان طراحی خاموش یا روشن نمود.
- Align To Grid : (ردیف بندی نسبت به شبکه) : اگر این گزینه انتخاب شده باشد، عناصر ابزارها فقط از ابتدای نقاط شبکه قابل رسم می باشند. در غیر این صورت عناصر را می توان در هر نقطه ای از فرم قرار داد.

- **چند ویژگی دیگر از Form :**

Width : عرض فرم بر حسب Twip

Height : ارتفاع فرم بر حسب Twip

Top : فاصله لبه بالای فرم از لبه بالای صفحه Desktop

Left : فاصله لبه چپ فرم از لبه چپ صفحه Desktop

Name : نام فرم. مقدار پیش فرض آن مانند Form1 است .ولی می توان نام آن را تغییر داد. این نام هنگام اشاره به فرم در زمان کد نویسی اهمیت دارد.

Control Box : توسط این ویژگی می توان تعیین نمود که پنجره فرم شکل منوی کنترل پنجره ویندوز باشد یا خیر. منوی کنترل پنجره، منویی است که با کلیک نمودن آیکون موجود روی نوار عنوان پنجره (قبل از عنوان پنجره) گشوده می شود. این منو شامل گزینه هایی مانند Move، Size، Minimize، Maximize و Close است.

Min Button و Max Button : این دو ویژگی تعیین می نمایند که آیا دکمه های حداقل و حداکثر نمودن اندازه پنجره وجود داشته باشند یا خیر. البته این دو دکمه زمانی فعال هستند که علاوه بر مقدار True، ویژگی Border Style نیز Sizeable باشد.

Icons : از این ویژگی برای تعیین نماد یا آیکون یک فرم استفاده می شود. با فعال نمودن این ویژگی، پنجره ای باز می شود که در آن می توان نام فایل آیکون با پسوند ico را تعیین نمود. تعدادی از این آیکونها در دایرکتوریی که VB را نصب نموده‌اید و در شاخه ای بنام Icon وجود دارد.

Mouse Pointer : توسط این ویژگی می توان شکل اشاره گر مأوس را بر روی فرم مورد نظر تعیین کرد. مقادیر نمونه برای این ویژگی عبارتند از :

مقدار	نوع شکل اشاره گر ماوس
Default:	مقدار پیش فرض، که شکل اشاره گر به محل آن و کنترل بستگی دارد.
Arrow:	فلش
Cross:	علامت بعلاوه یا صلیب
I Beam:	شکل I (جهت ورود اطلاعات)
Size:	علامت تغییر اندازه
Hourglass:	ساعت شنی
No Drop:	علامت ممنوع

WindowState: این ویژگی نوع نمایش یک فرم را تعیین می نماید. مقادیر آن عبارتند از :

Normal : حالت پیش فرض یا پنجره عادی .

Minimized : اندازه پنجره حد اقل می باشد.

Maximized : اندازه پنجره حداکثر می باشد.

۳۱ـ شیء دکمه فرمان (Command Buttons)

این کنترل، که یکی از رایج ترین کنترلهاست، برای شروع، قطع موقت یا قطع دائم یک زیر برنامه بکار می رود.

- **ویژگیهای دکمه فرمان :**

Appearance : برای انتخاب ظاهر تخت یا سه بعدی

Cancel : در صورتیکه این ویژگی Trueباشد، زدن دکمه Esc روی صفحه کلید سبب فعال شدن زیر برنامه مربوط به این دکمه فرمان می شود. دقت کنید که این ویژگی تنها برای یک دکمه روی فرم می تواند True باشد.

Caption: عنوان دکمه یا رشته متنی که روی دکمه نوشته می شود.

Default: در صورتیکه این ویژگی Trueباشد، زدن دکمه Enter روی صفحه کلید، سبب فعال شدن زیر برنامه مربوط به این دکمه فرمان می شود. دقت کنید که این ویژگی تنها برای یک دکمه روی فرم می تواند True باشد.

Font : برای تنظیم نوع، اندازه و سبک قلم.

- **رویدادهای دکمه فرمان :**

Click : رویدادی که فعال می شود، اگر کاربر بر روی دکمه فرمان کلیک کند یا کلیدهای میانبر آن را فشار دهد.

۳۲ـ جعبه های برچسب (Label Boxes)

جعبه برچسب، کنترلی است که برای نشان دادن متنی که کاربر مستقیماً نمی تواند تغییر دهد، بکار می رود. البته می دانیم که می توان متن یک جعبه بر چسب را هنگام اجرا و در پاسخ به یک رویداد تغییر داد.

- **ویژگیهای جعبه بر چسب :**

Alignment : برای تنظیم محل قرار گرفتن متن در جعبه

Appearance: برای انتخاب ظاهر تخت یا سه بعدی

Auto Size: در صورتیکه این ویژگی True باشد، جعبه بر چسب به اندازه متن داخل جعبه کوچک یا بزرگ می شود. برعکس، اگر این ویژگی False باشد، جعبه برچسب اندازه زمان طراحی خود را حفظ می کند. در این صورت اگر متن از جعبه بزرگتر باشد، قسمتی از

متن دیده نخواهد شد.

Border style : برای تعیین شکل حاشیه ها

Caption : عنوان یا متنی که داخل جعبه بر چسب نشان داده می شود.

Font : برای تنظیم نوع، اندازه و سبک قلم .

Ward Wrap : این ویژگی همراه با Autosize استفاده می شود. اگر AutoSize=True و WordWrap = True باشد، اندازه جعبه برچسب در جهت عمودی بزرگ می شود تا متن در آن جای بگیرد. در صورتیکهAuto Size = True و Word Wrap=False باشد، اندازه جعبه در جهت افقی امتداد می یابد تا متن در آن جای گیرد. در صورتیکه AutoSize = False باشد، مقدار WordWrap اهمیتی ندارد و اندازه جعبه تغییری نخواهد داشت.

- **رویدادهای جعبه بر چسب :**

Click : در صورتیکه کاربر بر روی یک برچسب کلیک کند، این رویداد فعال می شود.

Dblclick : این رویداد در صورت دابل کلیک کردن یک کاربر بر روی برچسب رخ می دهد.

۳۳ـ جعبه های متن (Text Boxes)

جعبه متن برای نشان دادن متن تعیین شده در فاز طراحی و یا وارد کردن اطلاعات توسط کاربر یا برنامه در زمان اجرا بکار می رود. متن نشان داده شده در جعبه متن قابل تغییر و ویرایش است.

- **ویژگیهای جعبه متن :**

Appearance : برای انتخاب ظاهر تخت یا سه بعدی

Border Style : برای تعیین نوع حاشیه ها

Font : برای انتخاب نوع، اندازه و سبک قلم

MaxLength : این ویژگی می تواند طول رشته متن در جعبه متن را محدود کند. در صورتیکه مقدار این ویژگی صفر باشد محدودیتی بر حسب طول رشته اعمال نمی گردد.

MultiLine : این ویژگی تعیین می کند که جعبه متن تنها یک خط یا چند خط را نمایش می دهد.

PasswordChar : کاراکتری که در این ویژگی تعیین می کنیم، کاراکتر رمز خواهد بود. بعبارتی متنی که در جعبه می نویسیم با این کاراکتر مخفی خواهد شد.

ScrollBars : برای تعیین نوع نوارهای لغزان

SelLength: طول رشته انتخاب شده (فقط قابل استفاده در زمان اجرا)

SelStart : شروع مکان متن انتخاب شده (فقط قابل استفاده در زمان اجرا)

SelText: متن انتخاب شده (فقط قابل استفاده در زمان اجرا)

Tag: برای ذخیره یک عبارت متنی

Text : متن نمایش داده شده

- **رویداد های جعبه متن :**

Change : هر گاه متن جعبه یا بعبارتی ویژگی Text تغییر کند، این رویداد فعال می شود.

LostFocus : هر گاه کاربر از جعبه متن خارج شود، فعال می گردد. این رویداد برای بررسی محتویات یک جعبه متن مناسب است.

KeyPress : هر گاه یک کلید روی صفحه کلید فشار داده شود، این رویداد فعال می شود. این رویداد برای به دام انداختن کلیدها (همانطور که قبلاً گفته شد) مناسب است.

- **متدهای جعبه متن :**

SetFocus : برای قرار دادن کرسر در یک جعبه متن مشخص. اگر برای یک جعبه متن این متد صدا زده شده باشد، کاربر نیازی ندارد که ابتدا این جعبه را انتخاب کرده و سپس شروع به نوشتن و وارد کردن اطلاعات بنماید. بلکه کرسر در این جعبه قرار داده شده است و کافیست کاربر مقادیر یا متن مورد نظر را بنویسید.

مثال :

```
txtExample.SetFocus
```

توجه: تا آنجا که ممکن است جعبه های متن را در برنامه کاهش دهید. هر گاه که جعبه متنی را در برنامه قرار می دهید، لازم است بعنوان برنامه نویس متن وارد شده را چک کنید تا با کد کردن برنامه شما مشکلی نداشته باشید. برای مثال اگر می خواهید کاربر عددی را در جعبه متن وارد کند، باید عدد بودن مقدار تایپ شده توسط کاربر را چک کنید.

۳٤ـ جعبه های انتخاب (Check Boxes)

جعبه های متن ابزار مناسب برای انتخاب از میان تعدادی کاندید است. از میان لیست می توان یک، چند، هیچکدام یا همه موارد را انتخاب کرد.

- **ویژگیهای جعبه انتخاب :**

Caption : این ویژگی برای تعیین متنی که در کنار جعبه انتخاب نوشته می شود، بکار میرود.

Font : برای تنظیم نوع، اندازه و سبک قلم .

Value : برای تعیین وضعیت انتخاب در جعبه :

0 یا VbUnchecked : به معنی عدم انتخاب

1 یا VbChecked : به معنی انتخاب

2 یا VbGrayed : به معنی خاکستری شدن جعبه

ـ رویدادهای جعبه انتخاب :

Click: هر گاه جعبه انتخاب کلیک شود این رویداد فعال می گردد. ویژگیValue بطور اتوماتیک توسط VB تعیین خواهد کرد.

۳۵ـ دکمه های انتخاب (Option Buttons)

این دکمه ها ابزارهای مناسبی هستند برای مواقعی که می خواهیم کاربر از میان چند مورد، یکی و تنها یکی را انتخاب نماید. بنابراین دکمه های انتخاب معمولاً بصورت گروهی وجود دارند که از میان این گروه، تنها یکی می تواند انتخاب شود.

- **ویژگیهای دکمه انتخاب :**

Caption : برای تعیین متنی که کنار دکمه انتخاب نوشته می شود.

Font : برای تنظیم نوع، اندازه و سبک قلم .

Value : اگر True باشد، یعنی دکمه انتخاب شده است. در غیر این صورت این ویژگی False است. تنها ویژگی Value یک دکمه در گروه دکمه های انتخابی می تواند True باشد.

- **رویدادهای دکمه انتخاب :**

Click : با کلیک کردن روی دکمه انتخاب این رویداد فعال می گردد. مقدار ویژگی Value بطور اتوماتیک توسط VB تغییر می کند.

۳۶ـ آرایه ها

- تا به اینجا ما همواره با متغیر های معمولی سروکار داشته ایم. این متغیرها هر کدام نشان دهنده یک واحد از حافظه بودند که نام مشخصی دارد. گاهی لازم است تعدادی از این خانه ها را داشته باشیم که تعریف آنها برایمان مطلوب نیست. برای مثال اگر بخواهیم نمرات ۱۰۰ دانش آموز را در متغیرها نگه داریم، باید ۱۰۰ متغیر با نام های مختلف استفاده کنیم.

  در VB امکاناتی برای تعریف و استفاده از آرایه ها وجود دارد. یک آرایه متغیری چند بعدی است.

- آرایه ها مشابه متغیرها تعریف میشوند. برای مثال برای تعریف کردن یک آرایه در سطح زیربرنامه که بتواند ۹ عدد صحیح را در خود ذخیره نماید:

```
Dim Items(9) as Integer
```

اگر بخواهیم مقادیر این آرایه پس از خروج از زیربرنامه از دست نرود، از کلمه Static استفاده مینماییم:

```
Static Items(9) as Integer
```

برای تعریف آرایه در سطح فرم یا مدول کافیست آرایه را در بخش General Declarations در پنجره کد تعریف نماییم:

```
Dim Items(9) as Integer
```

برای یک آرایه عمومی، در یک مدول آرایه را بصورت زیر تعریف میکنیم:

```
Global Items(9) as Integer
```

- شماره شاخص یا اندیس هر آرایه با صفرشروع شده، نهایتاً تا ابعاد مشخص شده برای آرایه میتواند بزرگ شود. برای مثال آرایه Items که در مثالهای فوق تعریف شد، دارای ۱۰ عنصر (یا خانه) است که عبارتند از Items(0) تا Items(9). از آرایه ها مانند سایر متغیرها استفاده می شود. تنها نکته قابل توجه این است که علاوه برنام، باید شماره شاخص یا اندیس را نیز ذکر کرد. برای آنکه در خانه Items(5) مقدار هفت را بنویسیم، دستور زیر لازم است :

```
Items(5)=7
```

- روش دوم تعریف آرایه، استفاده از حد بالا و پایین برای مشخص کردن دامنه تغییر شاخص یا اندیس آرایه است:

```
Dim A (1 to 10) as integer
```

در تعریف فوق شماره خانه های آرایه A از ۱ تا ۱۰ می تواند باشد (خانه صفر نداریم). دامنه تغییر شاخص لزوماً نباید از ۱ شروع شود:

```
Dim A (32 to 128) as single
```

بدیهی است که تعداد کل خانه های آرایه از فرمول زیر بدست می آید :

تعداد کل خانه ها = حد بالا – حد پایین + 1

- هنگامی که یک آرایه با دستور Dim تعریف می گردد، کلیه خانه های حافظه آن مقدار گذاری اولیه میشوند، به این مفهوم که کلیه خانه های حافظه آرایه های عددی با مقدار صفر و آرایه های رشته ای با رشته خالی (Null string) مقدار دهی میشوند.

- مثال: برنامه ای بنویسید که رشتهای را در جعبه متن با نام text1 بخواند و سپس با دستور print روی form1، تعداد دفعاتی که هر حرف در این رشته تکرار شده است را اعلام نماید.

آرایه ای به نام f تعریف میکنیم که تعداد دفعات تکرار ۲۶ حرف زبان انگلیسی را ذخیره نماید.

```
Dim f (1 to 20) as integer
Astr = text1.text
For I = 1 to Len (Astr)
        Achar= Ucase (Mid (astr, I,1))
        'Check if it is a letter
        If a Achar>="A" AND Achar <="Z" then
                Index= Asc (Achar)-Asc("A")+1
                F (index)=F(index) +1
        End if
Next
```

```
'Print  out the result
For I=1 to 26
        Form 1.print chr(I+ asc("A")-1); ":"; F(i),
Next
```

• مرتب سازی حبابی (Bubble Sort)

یکی از اعمالی که لازم است بتوانیم در آرایه ها انجام دهیم، مرتب سازی است. مرتب سازی میتواند بصورت صعودی (از کوچک به بزرگ) و یا نزولی (از بزرگ به کوچک) باشد؛ همچنین میتواند بر روی اعداد یا رشته ها صورت گیرد. یکی از مشهورترین و ساده ترین روشهای مرتب سازی، روش مرتب سازی حبابی است.

فرض نماییم پنج عدد بدون هیچ گونه ترتیب مشخصی درون پنج خانه یک آرایه قرار گرفته اند. طبق الگوریتم مرتب سازی حبابی، لازم است ابتدا دو عدد موجود در خانه های اول و دوم را با یکدیگر مقایسه نماییم. اگر عدد اول از عدد دوم بزرگتر باشد، باید جای آن دو عدد را با یکدیگر تعویض نماییم. در غیر این صورت تغییری انجام نمیگیرد.

۷۵	۱٤
۱٤	۷۵
۹ →	۹
۹٤	۹٤
۵۲	۵۲

چون عدد اول (۷۵) از عدد دوم (۱٤) بزرگتر است، جای آنها با هم تعویض گردیده است. حال همین عمل بر روی اعداد دوم و سوم، سوم و چهارم، چهارم و پنجم نیز انجام میگیرد

۱٤	۱٤	۱٤	۱٤
۷۵	۹	۹	۹
۹ →	۷۵ →	۷۵ →	۷۵
۹٤	۹٤	۹٤	۵۲
۵۲	۵۲	۵۲	۹٤

پس از اتمام این عملیات، بزرگترین عدد به آخرین سطر منتقل میشود. حال باید همین کار برای چهار عدد باقی مانده تکرار شود.

۱٤	۹	۹	۹
۹	۱٤	۱٤	۱٤
۷۵	۷۵	۷۵	۵۲
۵۲ →	۵۲ →	۵۲ →	۷۵
۹٤	۹٤	۹٤	۹٤

اگر چه در این مثال با همین دو تکرار اعداد به صورت صعودی مرتب شده اند، اما الگوریتم میگوید که در حالت کلی باید همین مراحل برای اعداد باقی مانده انجام دهیم.

بنابر این برای نوشتن کد برنامه، به دو حلقه نیاز داریم. حلقه اول به تعداد کل عناصر منهای یک، جهت انجام مراحل مقایسه و حلقه درونی جهت انجام مقایسه اعداد کنار هم به کار میرود. در حلقه درونی هر بار یک عدد از اعداد کنار رفته و حلقه کوچکتر می گردد.

با فرض اینکه اعداد در یک آرایه ۵ عنصری بنام array با تعریف زیر وجود دارند :

```
Const  Max=5
Dim array (Max) as integer
```

برنامه مرتب سازی حبابی بصورت زیر نوشته میشود :

```
For  I=1 to Max –1
        For  j=1 to Max –I
                If array (j)>array (j+1) then
                        Temp=array (j)
                        array(j)=array (j+1)
                        array(j+1)=Temp
                End if
        Next
Next
```

- اگر در برنامه اخیر به جای علامت بزرگتر (>) از علامت کوچکتر (<) استفاده شود، مرتب سازی به روش نزولی انجام خواهد گرفت.

- در زبان بیسیک مقایسه رشته ها بر اساس ترتیب کد اسکی آنها انجام می گیرد. بنابر این از برنامه قبل میتوان جهت مرتب سازی رشته ها نیز استفاده نمود .

۳۷- آرایه های چند بعدی

- فرض کنیم میخواهیم برای ۲۵ دانش آموز یک کلاس، سه نمره ریاضی، علوم، وهنر را در یک آرایه ذخیره نماییم. در این صورت میتوانیم یک آرایه دو بعدی بصورت زیر تعریف نماییم :

```
Dim A(1 to 25, 1 to 3) as single
```

در اینصورت در خانه A(i,j) نمره jام دانش آموز i ام قرار دارد.

مثال: برای ذخیره جدول ضرب در یک آرایه دو بعدی برنامه زیر را مینویسیم:

```
Dim A(10,10)as integer
For I=1 to 10
        For j=1 to10
                A(i,j)=i*j
        Next
Next
```

دقت کنید که خانه های A(0,j) و A(i,0) همگی دارای مقدار اولیه صفر بوده اند و این مقدار تغییر نمیکند.

- در زبان بیسیک امکان تعریف آرایه های حداکثر ۶۰ بعدی وجود دارد، ولی معمولا تا سه بعدی مورد استفاده قرار میگیرد.

تمرین: روی یک فرم ۵ عدد Textbox، یکLabel و یک Command قرار دهید.

الف) می خواهیم با کلیک بر روی دکمه فرمان، اعداد موجود درTextbox ها مرتب شده، روی Label نوشته شوند.

ب) برای آنکه این کار روی رشته های کاراکتری انجام شود چه باید بکنیم ؟

الف)

```
Private Sub Command _ Click()
        Dim F(1 to 5)as Integer
        F(1) = text1.text
        F(2) = text2.text
        F(3) = text3.text
        F(4) = text4.text
        F(5) = text5.text
        N=5
        For I=1 to N-1
                For j=1 to N-I
                        If F( j )> F (j+1) then
                                Temp=F (j )
                                F ( j )= F (j+1 )
                                F ( j+1)= F ( j )
                        End if
                Next
        Next
        For I=1 to N
                msg = msg & F( I ) & Space ( 3 )
        Next
        Labell.Caption = msg
End sub
```

ب) کافیست F را بصورت زیر تعریف کنیم :

```
Dim f (1 to 5) as String
```

۳۸- آرایه کنترلها

- گاهی مناسب است که آرایه ای از کنترلها را تعریف کنیم. برای دکمه های انتخاب عملاً همیشه اینطور عمل می نماییم .

- هنگامی که می خواهیم گروهی از کنترلها عملکرد مشابهی داشته باشند، یک راه ساده استفاده از آرایه هایی از کنترلهاست. همه رویدادهایی که برای یک کنترل موجود است, برای آرایه کنترلها نیز در دست می باشد. تنها تفاوت این است که یک پارامتر در رویداد موجود دارد که مشخص کننده کنترل انتخاب شده در آرایه است. بنابراین در حالتی که آرایه ای از کنترلها داریم، برای یکایک آنها کد نمی نویسیم، بلکه می توانیم یک کد برای همه آنها بنویسیم.

- یک مزیت دیگر در استفاده از آرایه کنترلها این است که می توان در زمان اجرا موردی را به آرایه اضافه کرد، یا آنرا حذف نمود. برای اینکار از دستورات Load و Unload استفاده می کنیم.

- **دو راه برای ایجاد آرایه کنترلها :**

۱. یک کنترل را ایجاد کنید و ویژگیهایش را تنظیم نمایید. کنترل را Copy کرده، سپس روی فرم Paste نمایید. پنجره ای باز می شود که از شما سوال می کند که می خواهید آرایه ای از کنترلها ایجاد نمایید یا خیر. جواب Yes بدهید و آرایه ساخته می شود.

۲. همه کنترلهایی را که می خواهید ایجاد کنید. نام دلخواه را به کنترل اول بدهید. سپس همان نام را به کنترل دوم اختصاص دهید. پنجره ای باز می شود و ایجاد آرایه ای از کنترلها را سوال می کند. جواب Yes بدهید و آرایه ایجاد می شود. سپس سایر موارد را به همان نام در آورید.

- پس از ایجاد آرایه کنترلها و نامگذاری آنها، هر عنصر آرایه بوسیله نام و شماره اندیس آن مشخص می شود. برای مثال اگر بخواهیم ویژگی Caption عنصر شماره شش آرایه ای از بر چسب ها بنام lblExample را تنظیم نماییم، بصورت زیر عمل می کنیم :

```
lblExample(6).Caption = “Hello”
```

۳۹- قابها (Frames)

- قابها وسیله ای برای گروه بندی کنترلهای مرتبط روی یک فرم می باشند. در صورت وجود دکمه های انتخاب، قابها روی چگونگی عملکرد آنها نیز اثر دارند.

- برای گروه بندی کنترلها در یک قاب، ابتدا قاب را رسم کنید. سپس کنترلهای دلخواه را درون قاب اضافه نمایید. در صورتیکه قاب را حرکت دهید، کنترلهای داخل آن همراه قاب حرکت خواهند کرد. پس از اضافه کردن یک کنترل در یک قاب می توانید آن را Copy کرده و برای ایجاد آرایه ای از کنترلها Paste نمایید. برای اینکار ابتدا روی کنترل کلیک کرده آنرا Copy کنید. سپس روی قاب کلیک کرده، Paste نمایید. از شما برای ایجاد آرایه ای از کنترلها سوال خواهد شد. جواب Yes بدهید.

- دقت کنید که عمل کردن بصورتهای زیر گروه بندی دلخواه را نتیجه نمی دهد:
  - رسم کنترلها در خارج از قاب و سپس Drag کردن آنها به داخل قاب.
  - Copy کردن کنترلها به داخل قاب.
  - رسم یک قاب در اطراف چند کنترل موجود روی فرم.

می توان در یک قاب، قاب دیگری را نیز رسم کرد.

- همانطور که گفته شد قابها روی چگونگی عملکرد دکمه های انتخاب اثر دارند. دکمه های انتخاب درون یک قاب بصورت یک گروه عمل می کنند و مستقل از دکمه های انتخاب در قابهای دیگر هستند. دکمه های انتخاب روی فرم نیز(که در هیچ قابی واقع نمی شوند) مستقل از سایر گروه ها میباشند. بعبارتی یک فرم خود یک قاب است .

- توجه نمایید که یک گروه مستقل از دکمه های انتخاب بوسیله مکان فیزیکی آنها در قابها مشخص می شود و نه با نام گذاری آنها. بعبارتی یک آرایه از دکمه های انتخاب تنها به علت آرایه بودن بصورت یک گروه مستقل عمل نمی کند, بلکه عمل کردن آن بصورت یک گروه مستلزم آن است که تنها آرایه از دکمه های انتخاب در یک قاب یا فرم باشد.

- **ویژگیهای قاب:**

Caption عنوانی که در بالای قاب ظاهر می شود.

Font برای تعیین نوع و اندازه قلم .

٤٠- جعبه های لیست (List Boxes)

- یک جعبه لیست، لیستی از موارد را نمایش می دهد و کاربر می تواند یک یا تعدادی از آنها را انتخاب نماید. اگر تعداد موارد لیست از تعدادی که می تواند نشان داده شود، بیشتر باشد نوار لغزان بطور خودکار اضافه خواهد شد.

- **ویژگیهای جعبه لیست :**

Appearance برای تعیین ظاهر تخت یا سه بعدی .

List آرایه ای از موارد لیست.

ListCount تعداد موارد لیست.

ListIndex شماره آخرین مورد انتخاب شده در لیست. در صورتیکه موردی انتخاب نشده باشد، ListIndex برابر ١- است.

MultiSelect برای تعیین نحوه انتخاب .
(0. انتخاب چند مورد ممکن نیست.)
(١. انتخاب چند مورد مجاز است.)
(٢. انتخاب گروهی مجاز است.)

Selected آرایه ای از عناصر True یا False. در صورتیکه موردی از لیست انتخاب شده باشد، عنصر متناظر آن در این آرایه True خواهد بود.

Sorted اگر این ویژگی True باشد، موارد لیست به ترتیب Ascii مرتب خواهند شد. در غیر این صورت ترتیب قرار گرفتن موارد در لیست به ترتیب اضافه شدن به لیست خواهد بود.

Text متن آخرین مورد انتخاب شده .

- **رویدادهای جعبه متن :**

Click در صورتی فعال می شود که بر روی موردی در لیست کلیک شود.

DblClick در صورتی روی می دهد که موردی در لیست، دابل کلیک گردد.

- **متدهای جعبه متن :**

AddItem برای اضافه کردن موردی به لیست.

Clear برای حذف همه موارد لیست.

RemoveItem برای حذف موردی از لیست با اندیس مشخص.

مثال :

```
LstExample.AddItem “Item1”
LstExample.Clear
LstExample. RemoveItem 5
```

- خط اول موردی با متن ‘Item1‘ را به انتهای لیست اضافه می کند.
- خط دوم کل لیست را پاک می کند.
- خط سوم، پنجمین مورد از لیست، یا بعبارتی LstExample.List (5) را حذف می کند.

- موارد موجود در لیست یک جعبه لیست معمولاً در فرایند Form_ Load مقدار اولیه داده می شوند. بهتر از قبل از تعیین موارد لیست، از متد Clear برای خالی کردن لیست استفاده شود.

٤١- جعبه های Combo (Combo Box)

یک جعبه Combo بسیار مشابه یک جعبه لیست است. تنها تفاوت این است که یک جعبه Combo دارای یک جعبه متن در بالای لیست است و کاربر می تواند یک مورد را انتخاب نماید یا در مواقعی مورد جدیدی را تایپ کند.

- **ویژگیهای جعبه Combo :**

Appearance : برای تعیین ظاهر تخت یا سه بعدی .

List : آرایه ای از موارد لیست.

ListCount : تعداد موارد لیست.

ListIndex : شماره آخرین مورد انتخاب شده در لیست. اگر موردی انتخاب نشده باشد List Index=-1 است.

Sorted: اگر True باشد، موارد لیست به ترتیب Ascii مرتب می شوند.

Style: برای تعیین شکل و نحوه های عملکرد جعبه Combo

; DropDown Combo+ Text Box : Style=0

در این حالت کاربر می تواند انتخاب متفاوتی داشته باشد.

; Simple Combo + Text Box : Style=1

در این حالت نیز کاربر می تواند انتخاب متفاوتی داشته باشد. لازم است اندازه پیش فرض جعبه تغییر داده شود تا قسمت لیست ظاهر گردد.

; DropDown Combo : Style=2

کاربر حق انتخاب دیگری ندارد.

Text : متن آخرین مورد انتخاب شده در لیست.

- **رویدادهای جعبه Combo**

Click در صورتی فعال می شود که بر روی موردی در لیست کلیک شود.

DblClick در صورتی روی می دهد که موردی در لیست، دابل کلیک گردد.

- **متدهای جعبه Combo**

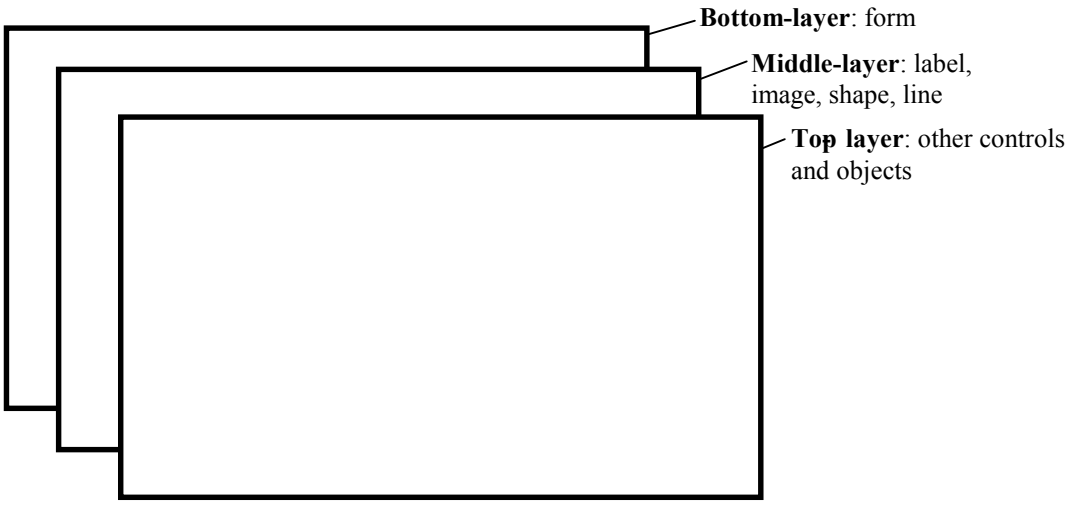
Add Item برای اضافه کردن موردی به لیست.

Clear برای حذف همه موارد لیست.

Remove Item برای حذف موردی از لیست با اندیس مشخص.

٤٢- لایه های نمایش (Display layers)

- کنترلهایی که روی یک فرم قرار میگیرند، لزوماً روی یک سطح یا لایه نمایش واقع نمیشوند، بلکه ئیک فرم از سه لایه تشکیل شده است. اطلاعاتی که مستقیماً روی فرم نشان داده میشوند (مثلاً با متد Print یا متدهای گرافیکی) روی پایین ترین لایه قرار میگیرند. اطلاعات روی جعبه های برچسب، جعبه های شکل، ابزار رسم خط، و ابزار رسم اشکال روی لایه میانی نشان داده میشوند. سایر کنترلها و اشیاءِ روی لایه بالایی فرم واقع میگردند.



- باتوجه به آنچه گفته شد، هنگام قرار گرفتن کنترلها روی فرم باید مراقب باشیم که کنترلی را نپوشانیم و کنترل موردنظر توسط سایر کنترلها پوشانده نشود. برای مثال یک دکمه فرمان میتواند متنی که روی فرم پرینت میشود را بپوشاند. همچنین اشکال رسم شده توسط کنترل شکل توسط همه کنترلها بجز جعبه شکل پوشانده میشود.

- ترتیب قرار گرفتن کنترلها در یک لایه به ویژگی Zorder آنها بستگی دارد. بعبارتی اگر دو کنترل در یک لایه با هم همپوشانی داشته باشند، این ویژگی تعیین میکند که کدام یک روی دیگری واقع شود. ترتیب رسم کنترلها و اضافه کردن آنها به فرم مقدار پیش فرض این ویژگی را تعیین میکند. بعبارتی کنترلی که دیرتر اضافه شده باشد، روی کنترلی که قبلاً اضافه شده قرار میگیرد. اما با کلیک راست روی کنترلها و انتخاب دو مورد زیر میتوان ترتیب فوق را تغییر داد:

Bring to Front: آوردن کنترل به روی کنترلهای دیگر

Send to Back: بردن کنترل به زیر کنترلهای دیگر

دقت کنید که دو دستور فوق ترتیب کنترلها را تنها در یک لایه تغییر میدهد. کنترلها یی که در لایه بالاتری قرار دارند، همیشه روی کنترلهایی که در لایه زیرتری واقع میشوند، نشان داده خواهند شد.

٤٣- ابزار رسم خط (Line Tool)

- بکمک این کنترل میتوان پاره خطهایی با طول، ضخامت، و رنگهای مختلف روی فرم رسم نمود. این کنترل برای زیباترکردن فرم استفاده میشود.

- **ویژگیهای کنترل رسم خط:**

BorderColor: برای تعیین رنگ خط

BorderStyle: برای تعیین شکل خط. خطها میتوانند بصورت توپر، خط فاصله، نقطه چین و غیره رسم شوند.

BorderWidth: برای تعیین ضخامت خط

- ابزار رسم خط دارای رویداد یا متدی نمیباشد.
- ازآنجاییکه کنترل خط در لایه میانی فرم واقع میشود، همه کنترلها بجز ابزار رسم شکل، جعبه برچسب و جعبه شکل آنرا خواهند پوشاند.

٤٤- ابزار رسم شکل (Shape Tool)

- ابزار رسم شکل برای ایجاد شکلهای دایره، بیضی، مربع، مستطیل، مربع، و مستطیل با گوشه های گرد میتواند بکار رود. این اشکال را میتوان با رنگها و طرحهای متنوع در فرم قرار داد.

- **ویژگیهای کنترل رسم شکل:**

BackColor: برای تعیین رنگ زمینه شکل وقتی که ویژگی FillStyle دارای مقدار Solid (یا توپر) نباشد.

BackStyle: برای تعیین شفاف یا مات بودن زمینه

BorderColor: برای تعیین رنگ حاشیه شکل

BorderStyle: برای تعیین شکل حاشیه. حاشیه میتواند شفاف، توپر، خط فاصله، نقطه چین، یا ترکیبی از اینها باشد.

BorderWidth: برای تعیین ضخامت حاشیه

FillColor: برای تعیین رنگ داخل شکل

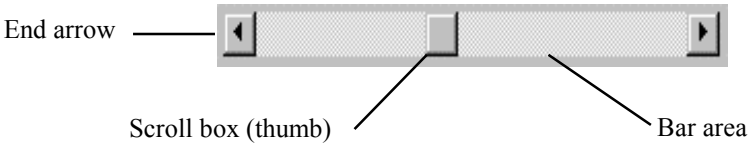
FillStyle: برای تعیین الگوی رنگ آمیزی در داخل شکل، برای مثال توپر، ضربدر، شفاف، مات، و غیره.

Shape: برای تعیین شکل، برای مثال مربع (Square)، مستطیل (Rectangle)، دایره، و غیره.

- ابزار رسم شکل، رویداد یا متدی ندارد.
- ابزار رسم شکل در لایه میانی قرار دارد و توسط کنترلهای لایه فوقانی پوشانده میشود.

۴۵- نوارهای لغزان افقی و عمودی (Horizontal and Vertical Scroll Bars)

- نوارهای لغزان در برنامه های ویندوز بسیار مورد استفاده قرار میگیرند. این نوارها ابزار مناسبی برای حرکت در میان لیستی از اطلاعات میباشند.
- هر نوار لغزان شامل سه ناحیه میباشد که میتوانند کلیک یا دراگ شوند تا مقدار نوار لغزان را تغییر دهند. کلیک روی یک فلش انتهایی (End Arrow) مقدار نوار لغزان را به مقدار کم و کلیک روی ناحیه نواری نوار لغزان (Bar Area)، مقدار نوار را به میزان بیشتر تغییر میدهد. حرکت دادن یا دراگ کردن جعبه نوار لغزان (Scroll Box) مقدار نوار را بصورت پیوسته تغییر خواهد داد. بکمک ویژگیهای نوار لغزان میتوان چگونگی عملکرد آن را تعیین نمود. مکان جعبه نوار لغزان، اطلاعات خروجی یک نوار است.



- **ویژگیهای نوار لغزان:**

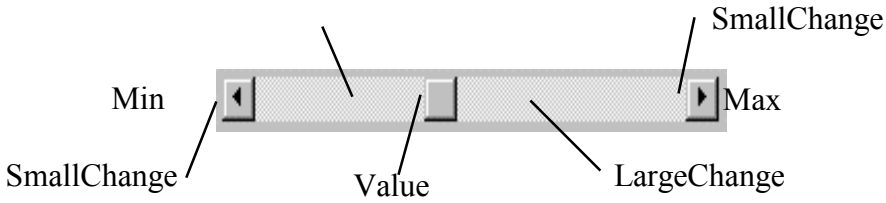
LargeChange: مقداری که با کلیک کردن روی ناحیه نواری نوار لغزان از ویژگی Value آن کم میشود یا به آن اضافه میگردد.

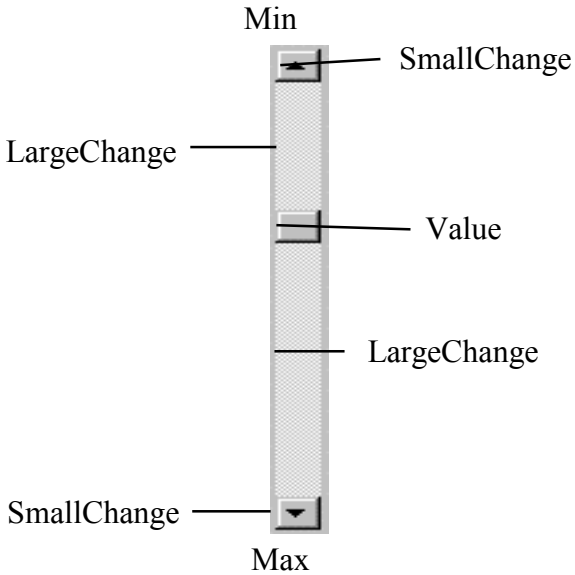
Max: مقدار نوار لغزان در انتهای سمت راست نوار لغزان افقی یا انتهای پایینی نوار لغزان عمودی. این مقدار میتواند از ۳۲،۷۶۷- تا ۳۲،۷۶۷ انتخاب گردد.

Min: مقدار نوار لغزان در انتهای سمت چپ نوار لغزان افقی یا انتهای بالایی نوار لغزان عمودی. این مقدار میتواند از ۳۲،۷۶۷- تا ۳۲،۷۶۷ انتخاب گردد.

SmallChange: مقداری که با کلیک کردن روی فلشهای انتهایی نوار لغزان از ویژگی Value آن کم میشود یا به آن اضافه میگردد.

Value: مکان فعلی جعبه نوار لغزان در ناحیه نوار. اگر به این ویژگی مقدار دهید، جعبه نوار لغزان به مکان ذکر شده حرکت خواهد کرد.





- **نکات قابل توجه درباره نوارهای لغزان:**

۱- توجه کنید که دو ویژگی Min و Max مشخص کننده مقدار ویژگی در دو انتهای نوارند و لزوماً نشان دهنده مقادیر مینیمم و ماکزیمم نمیباشند. بعلاوه نیازی نیست که مقدار Max از مقدار Min بزرگتر باشد. علامت دو ویژگی SmallChange و LargeChange بطور خودکار توسط VB طوری تعیین خواهد شد که حرکت بطور صحیح بین Min وMax انجام گیرد.

۲- دقت کنید اگر مقدار Min، Max، و Value را در کد برنامه تغییر میدهید، مقدار Value همواره بین Min و Max قرار گیرد. درغیر اینصورت پیغام خطا دریافت خواهید نمود.

- **رویدادهای نوار لغزان:**

Change: رویدادی که با هرگونه تغییر در مکان جعبه نوار لغزان روی میدهد. میتوان از این رویداد برای خواند مقدار ویژگی Value پس از هر تغییر در نوار لغزان استفاده کرد.

Scroll: رویدادی که با جابجا کردن جعبه نوار لغزان اتفاق می افتد.

۴۶- جعبه های تصویر PICTURE BOXES

- جعبه تصویر امکان نشان دادن اطلاعات گرافیکی را روی فرم فراهم می آورد. این کنترل برای محیط دینامیک، برای مثال نشان دادن انیمیشن، مناسب است .
- جعبه های تصویر در بالاترین لایه نمایش روی فرم نشان داده می شوند. این کنترل بسیار مشابه خود فرم است و بسیاری از ویژگیهای فرم را نیز دارد .

- **ویژگیها :**

Auto Size: اگر True باشد، جعبه بطور اتوماتیک هم اندازه تصویر خود میشود .

Font: برای تنظیم اندازه و سبک در جعبه تصویر و هر چه روی آن print می شود .

Picture: برای تعیین تصویری که در جعبه نشان داده می شود .

- **رویدادها :**

Click: وقتی که روی جعبه کلیک شود، این رویداد رخ می دهد .

Dbl Click: با دابل کلیک روی جعبه، این رویداد رخ می دهد .

- **متدها :**

Cls: جعبه تصویر را پاک می کند .

Print: اطلاعات مورد نظر را روی جعبه تصویر پرینت می کند .

برای مثال :

```
PicExample.Cls
```

جعبه تصویری با نام PicExample پاک می کند .

```
PicExample.Print  "a Picture box "
```

رشته متن ذکر شده را روی جعبه تصویر چاپ می کند .

- **تابع LoadPicture:**

یکی از توابع مهم هنگام استفاده از جعبه های تصویر، تابع LoadPicture میباشد این تابع می تواند ویژگی Picture جعبه تصویر را در زمان اجرا تنظیم نماید .

برای مثال :

PicExample.Picture = LoadPicture(“c:\Sample.bmp “)

دستور فوق فایل گرافیکی که در مسیر c :\Sample.bmp موجود است را باز کرده و می خواند. سپس آنرا در ویژگی Picture جعبه تصویر PicExample قرار میدهد. توجه نمایید که مسیر فایل گرافیکی باید صحیح و کامل باشد، در غیر این صورت برنامه پیغام خطا خواهد داد .

پنج نوع فایل گرافیکی را می توان در یک جعبه تصویر نشان داد :

Bitmap	تصویری که بوسیله پیکسلها ارائه می شود و بصورت مجموعه ای از بیت ها که هر بیت متناظر با یک پیکسل است ذخیره می گردد. معمولا پسوند bmp. دارد ودر اندازه واقعی نشان داده می شود .
Icon	نوع خاصی از فایل Bitmap که اندازه اش حداکثر ۳۲*۳۲ پیکسل است. پسوند ico. دارد و در اندازه واقعی نشان داده می شود.
Metafile	فایلی که تصویر را بصورت مجموعه ای از موجودیت های گرافیکی (مانند: خط، دایره، چند ضلعی) ذخیره می کند. این نوع از تصویر هنگام تغییر اندازه کمتر خراب می شود، دارای پسوند wmf. است و تصاویر این نوع، به اندازه جعبه تصویر نشان داده می شوند.
JPEG	نوع JPEG (گروه اتحاد خبرگان فتوگرافیک) نوعbitmap فشرده ای است که رنگهای هشت و ۲٤ بیتی را پشتیبانی می کند. این نوع در اینترنت بسیار معمول است، پسوند jpg. دارد و در اندازه واقعی نشان داده می شود.
GIF	نوع GIF (فرمت تبادل گرافیکی) نوع فشرده ای است که توسط Comu Serve تهیه شده است. تا ۲۵۶ رنگ را پشتیبانی می کند و در اینترنت معمول است. پسوند gif. دارد ودر اندازه واقعی نشان داده میشود.

٤٧- جعبه شکل Image Box

- جعبه شکل بسیار شبیه جعبه تصویر است و مانند آن امکان نشان دادن اطلاعات گرافیکی را فراهم می آورد. این کنترل برای مواقع استاتیک و مواقعی که تغییری در نحوه نمایش تصویر نیاز نیست مناسب تر است.

- جعبه های شکل در لایه میانی نمایش روی فرم نشان داده میشوند. بنابراین جعبه های تصویر و سایر کنترلها میتوانند آنها را بپوشانند. تصاویر موجود در این کنترلها را میتوان با ویژگی Stretch تغییر اندازه داد.

- **ویژگیهای جعبه شکل**

Picture	برای تعیین فایل گرافیکی که در جعبه شکل نشان داده می شود.
Stretch	اگر False باشد، جعبه متن به اندازه تصویر در می آید. اگر True باشد، تصویر به اندازه جعبه متن در خواهد آمد.

- **رویدادهای جعبه شکل**

Click هنگام کلیک روی جعبه شکل روی می دهد.
DblClick هنگام دابل کلیک روی جعبه شکل روی می دهد.

- جعبه شکل متدی ندارد، اما میتوان تابع LoadPicture را برای آن بکار برد. این تابع مانند جعبه تصویر بکار میرود و همان نوع فایلها، Bitmap(.bmp)، metafile (.wmf)، icon(.icon)، فایلهای GIF(.gif) و فایلهای JPEG(.jpg) را پشتیبانی می نماید. اگر ویژگی Stretch برابر True باشد، هر پنج نوع تصویر گرافیکی به اندازه جعبه شکل در خواهند آمد. نوعهای Metafile، GIF، JPEG هنگام تغییر اندازه زیباتر می شوند.

۴۸- جعبه لیست درایو Drive List Box

- جعبه لیست درایو به کاربر اجازه می دهد که در زمان اجرا از لیست درایوهای موجود، درایوی را انتخاب نماید. درایوهای موجود در یک جعبه Combo بطور خود کار نشان داده می شوند و نیازی به کد نویسی نیست.

- **ویژگیها جعبه لیست درایو**

Drive نام درایوی که انتخاب شده است.

- **رویدادهای جعبه لیست درایو**

Change هنگامی که کاربر یا برنامه انتخاب درایو را تغییر دهند، این رویداد فعال می گردد.

۴۹- جعبه لیست دایرکتوری Directory List Box

- جعبه لیست دایرکتوری لیست مرتب و سلسله مرتبی از شاخه ها و زیر شاخه های دیسک را نشان می دهد. ساختار درختی شاخه ها در یک جعبه لیست بدون نیاز به کد نویسی نشان داده می شود.

- **ویژگیهای جعبه لیست دایرکتوری :**

Path مسیر دایرکتوری فعلی

- **رویدادهای جعبه لیست دایرکتوری :**

Change با تغییر انتخاب دایرکتوری، این رویداد فعال می گردد.

۵۰- جعبه لیست فایل File List Box

- جعبه لیست فایل، فایلهایی که در مسیر تعیین شده توسط ویژگی Path آن وجود دارد یافته، در زمان اجرا آنها را لیست می کند. می توان نوع فایلهایی که نشان داده می شوند را محدود کرد.

- **ویژگیهای جعبه لیست فایل**

FileName نام آخرین فایلی که انتخاب شده است.

ListCount تعداد فایلهایی که انتخاب شده اند.

List آرایه ای از نام فایلهای موجود در جعبه لیست

MultiSelect برای مجاز کردن انتخاب چند گانه در لیست

Path مسیر دایرکتوری فعلی

Pattern در این ویژگی می توان رشته ای برای محدود کردن نوع فایلهایی که نشان داده می شوند، تعیین نمود. در این ویژگی می توان از کاراکترهای جایگزین * ،? استفاده کرد. برای مثال استفاده از رشته *dat، باعث می شود تنها فایلهایی که پسوند dat دارند در جعبه لیست شوند.

- **رویدادهای جعبه لیست فایل :**

DblClick با دابل کلیک روی نام یک فایل فعال می گردد.

PathChange با تغییر مسیر یک جعبه لیست فایل روی می دهد.

۵۱- هماهنگی جعبه های لیست درایو، دایرکتوری و فایل

معمولا سه جعبه لیست درایو، دایرکتوری و فایل با هم بکار می روند تا بتوان نام فایلی را در محلی از دیسکت بطور دقیق تعیین نمود. بنابراین لازم است این سه با هم هماهنگ باشد تا نام فایلهای نمایش داده شده مربوط به درایو و دایرکتوری صحیح باشد.

هر گاه انتخاب درایو تغییر کند، رویداد Change جعبه لیست درایو فعال می گردد و لازم است مسیر جعبه دایرکتوری بهنگام در آورده شود. برای مثال اگر نام جعبه درایو drvExample و نام جعبه دایرکتوری dirExample باشد، کد زیر لازم است :

```
DirExample.Path = drvExample.Drive
```

هرگاه انتخاب دایرکتوری تغییر کند، رویداد Change جعبه دایرکتوری روی می دهد و مسیر جعبه فایل باید تغییر کند تا فایلهای دایرکتوری انتخاب شده، لیست شوند. در صورتیکه نام جعبه فایل fileExample باشد، کد زیر لازم است :

```
FileExample.path = dirExaple.path
```

پس از آنکه درایو، دایرکتوری و نام فایل توسط کاربر انتخاب گردید، لازم است یک رشته متنی برای تعیین مسیر کامل فایل آماده گردد. در این رشته نام درایو، دایرکتوری و فایل وجود دارد. بعلاوه وجود کاراکتر (\ ) backslash نیز حائز اهمیت است. توجه نمایید که در صورتیکه فایل در مسیر ریشه وجود داشته باشد، نام مسیر به (\ ) منتهی می شود، حال آنکه در صورتیکه در دایرکتوری باشد، نام مسیر (\ ) ندارد. بنابراین می توان کد زیر را برای بدست آوردن مسیر کامل فایل نوشت :

```
Dim YourFile as String
If      Right (filExample.Path , 1) = " \ " Then
        YourFile = filExample.Path + filExample.FileName
Else
YourFile = filExample.Path + " \ " + filExample.FileName
End If
ImgExample.Picture = LoadPicture (Your File)
```

توجه کنید که در تعیین مسیر کامل فایل، تنها از ویژگیهای جعبه فایل استفاده کرده ایم. ویژگیهای جعبه درایو و دایرکتوری تنها برای اعمال تغییرات در جعبه لیست فایل مورد استفاده قرار می گیرند.

۵۲- جعبه های دیالوگ عمومی Common Dialog Boxes

- جعبه های لیست درایو، دایرکتوری و فایل برای استفاده های خاص و اختصاصی بکار می روند. در حالت های کلی، دو روش عمومی برای دسترسی به فایل در برنامه های ویندوز وجود دارد. این دو روش بنامهای Open File و Save File در ویژوال بیسیک آماده به استفاده هستند. برای بکار گرفتن پنجره های استاندارد جهت کاربردهای عمومی، ویژوال بیسیک مجموعه ای از جعبه های دیالوگ عمومی را ارائه می دهد، که در آنها پنجره های دیالوگ Save As و Open نیز موجود است. این پنجره ها در همه برنامه های ویندوز به چشم می خورند و برای کار بر آشنا هستند.
- برای استفاده از پنجره های دیالوگ عمومی لازم است در منوی Project گزینه Components را انتخاب کرده، از لیست کنترلها،

Microsoft Common Dialog Control

را تیک زده و OK کنید. سپس می توانید از جعبه ابزار این کنترل را به فرم اضافه نمایید.

- اگر چه در زمان طراحی کنترل جعبه دیالوگ های عمومی روی فرم دیده میشود، این کنترل در زمان اجرا مشاهده نخواهد شد. بعلاوه محل باز شدن پنجره دیالوگ نیز قابل تنظیم نیست. می توان بکمک متدهای زیر تعیین کرد که کدام دیالوگ باز شود :

متد	پنجره دیالوگ عمومی
ShowOpen	پنجره دیالوگ Open (بازیابی)
ShowSave	پنجره دیالوگ Save As (ذخیره)
ShowColor	پنجره دیالوگ Color (رنگ)
ShowFont	پنجره دیالوگ Font (قلم)
ShowPrinter	پنجره دیالوگ Printer (چاپگر)

برای آنکه پنجره Open توسط یک کنترل دیالوگ عمومی بنام cdlExample باز شود، دستور زیر لازم است:
CdlExaple.ShowOpen

- پس از آنکه کاربر پنجره دیالوگ Open را به طریقی ببندد، اجرای دستورات پس از این دستور انجام خواهد گرفت. توجه کنید که پنجره های دیالوگ عمومی System Modal هستند. بعبارتی تا به آنها جواب ندهید نمی توانید به برنامه خود یا برنامه دیگری ادامه دهید.

در ادامه استفاده از پنجره های Save As و Open را بیشتر بررسی خواهیم کرد.

• پنجره دیالوگ عمومی Open

پنجره Open به کاربر امکان می دهد که نام فایلی را جهت باز کردن آن مشخص نماید. در اینجا هدف فقط تعیین نام فایل است و کار کردن با فایلها مورد نظر نیست. برای نشان دادن پنجره از متد ShowOpen استفاده میکنیم.

• ویژگیهای پنجره دیالوگ Open

Cancel Error	اگر True باشد و در صورتیکه در پنجره دیالوگ دکمه Cancel کلیک شود، یک خطا ایجاد می شود. میتوان از روشهای بررسی خطا برای تشخیص انتخاب Cancel بهره گرفت.
DialogTitle	عنوان بالایی پنجره Open که مقدار پیش فرض آن نیز Open است.
FileName	بکمک این ویژگی میتوان نام فایل اولیه که با بازشدن پنجره نشان داده می شود را تنظیم نمود. بعلاوه پس از بسته شدن پنجره، این ویژگی نام فایل انتخاب شده را مشخص می نماید.
Filter	این ویژگی برای محدود کردن فایلهایی که در جعبه لیست فایلها نشان داده می شوند، بکار می رود. برای مثال تعیین فیلتر بصورت Bitmaps (*.bmp) | *.bmp سبب می شود که تنها فایلهای با پسوند bmp نشان داده شوند و در قسمت Files of type عبارت Bitmaps(*.bmp) نوشته شود.

FilterIndex اگر چند فیلتر تعیین شده باشد، شماره فیتلر پیش فرض را مشخص می کند. مقدار اولیه این ویژگی یک است.

- **پنجره دیالوگ عمومی Save As**

پنجره Save As هنگام ذخیره سازی، امکان تعیین نام فایل و محل ذخیره آن را فراهم می آورد. برای نشان دادن این پنجره میتوان از متد Show Save استفاده نمود.

- **ویژگیهای پنجره Save As :**

CancelError اگر True باشد و در صورتیکه در پنجره دیالوگ دکمه Cancel کلیک شود، یک خطا ایجاد می شود. میتوان از روشهای بررسی خطا برای تشخیص انتخاب Cancel بهره گرفت.

DefaultExt برای تنظیم پسوند پیش فرض فایل، اگر پسوندی تعیین نشده باشد.

DialogTitle عنوان بالایی پنجره که مقدار پیش فرض آن نیز Save As است.

FileName بکمک این ویژگی میتوان نام فایل اولیه که با بازشدن پنجره نشان داده می شود را تنظیم نمود. بعلاوه پس از بسته شدن پنجره، این ویژگی نام فایل انتخاب شده را مشخص می نماید.

Filter این ویژگی برای محدود کردن فایلهایی که در جعبه لیست فایلها نشان داده می شوند، بکار می رود. برای مثال تعیین فیلتر بصورت

Bitmaps (*.bmp) | *.bmp

سبب می شود که تنها فایلهای با پسوند bmp نشان داده شوند و در قسمت Files of type عبارت Bitmaps(*.bmp) نوشته شود.

FilterIndex اگر چند فیلتر تعیین شده باشد، شماره فیتلر پیش فرض را مشخص می کند. مقدار اولیه این ویژگی یک است.

۵۳- زیر برنامه ها و توابع

- ویژوال بیسیک یک زبان برنامه نویسی ساخت یافته (structured) است. خصوصیت اصلی برنامه نویسی ساخت یافته, تفکیک و جداسازی قسمت های مختلف برنامه است. در برنامه های ساخت یافته می توان عملیات جانبی را در بخش های مجزا و تفکیک شده قرار داد. در موقع لزوم این عملیات فراخوانی (Call) یا صدا زده می شوند و پس از انجام عملیات مزبور, مجدداً به برنامه صدا زننده بر می گردند. به این بخشهای تفکیک شده که بخشی از برنامه بوده و عمل خاصی را انجام می دهند, زیر برنامه یا روال فرعی (Subroutine or Procedure) می گویند.

- حسن استفاده از زیر برنامه ها افزایش خوانایی برنامه و در نتیجه، سادگی خطایابی و اعمال تغییرات در آن می باشد. بعلاوه به این وسیله می توان کار برنامه نویسی را بین چند برنامه نویس تقسیم کرد.

- دسته ای از زیر برنامه ها، مقداری را به بخش فراخواننده بر می گردانند. به این زیر برنامه ها، تابع (Function) می گویند. بعبارتی، تابع زیر برنامه ای است که پس از انجام عملیات ویژه و پردازش اطلاعات، نتیجه را برمی گرداند.

- برای تعریف کردن یک زیر برنامه یا تابع بهتر است به یک Module رفته، عبارت زیر را بنویسیم :

تعریف زیر برنامه :

```
Sub  نام زیر برنامه
End Sub
```

تعریف تابع:

```
Function نام تابع
End Function.
```

- مثال: زیر برنامه ای بنویسید که ۱۰ عدد ستاره روی فرم یک پرینت کند.

```
Sub Star
    Form1.Print String( 10,"*")
End Sub.
```

هر گاه لازم باشد در برنامه از این زیر برنامه استفاده کنیم، می توانیم آن را به یکی از دو صورت زیر صدا کنیم :

```
Call Star
Star      یا
```

- می توان برای زیربرنامه ها و توابع پارامترهای ورودی در نظر گرفت. برای استفاده از پارامترها, لازم است آنها را درون یک جفت پرانتز و در جلوی نام زیر برنامه یا تابع قرار دهیم.

- مثال : زیر برنامه ای بنویسید که به تعداد دلخواه ستاره روی Form1 پرینت کند.

```
Sub Nstar (No as integer)
    If  No>0 then
        Form1.Print  String (No, "*")
   end if
End Sub.
```

برای چاپ ۱۰ ستاره یا ۵۰ ستاره می توانیم دستورات زیر را در برنامه بنویسیم.

```
Call Nstar (10)
Nstar  50
```

- اگر در هنگام فرخوانی یک زیر برنامه یا تابع به جای مقدار، از یک متغیر استفاده نمایید، هر گونه تغییر در زیر برنامه بر روی آرگومان آن، سبب تغییر در متغیر اصلی می شود. به این حالت فراخوانی با مرجع یا آدرس می گویند.

برای مثال :

Nstar x

در صورتیکه از یک مقدار در فراخوانی استفاده شود، اصطلاحاً فراخوانی با مقدار روی داده است. برای مثال :

Nstart 10

یا Nstart x+0

- پیش فرض تعریف متغیرهای یک زیر برنامه یا تابع، بصورت فراخوانی با آدرس است. ولی می توان متغیرها را بصورتی تعریف کرد که فراخوانی تنها با مقدار صورت گیرد.

Sub Nstar (NoStar as Integer)

یا

Sub Nstar (ByRef NoStar as Integer)

در صورتیکه متغیرNoStar به یکی از دو صورت فوق تعریف شود، مجاز هستیم NoStar را در زیر برنامه تغییر دهیم و این تغییر روی متغیری که در صدا زدن زیر برنامه بکار می بریم، موثر است. برعکس، در تعریف زیربرنامه بصورت زیر:

Sub Nstar (ByVal Nostar as Integer)

تعریف متغیر NoStar با عبارت ByVal به این معناست که تنها مقدار متغیر در زیر برنامه استفاده می شود.

- توجه: برای استفاده از یک متغیر در صدا زدن یک زیر برنامه یا تابع، نوع متغیر باید با تعریف متغیر متناظر در تعریف زیر برنامه یا تابع همخوانی داشته باشد. برای مثال اگر زیر برنامه بصورت زیر تعریف شده است :

Sub Nstar (Nostar as Integer)

برای صدا کردن برنامه با متغیر x :

Dim x as Integer
Call Nstar (x)

در صورتیکه متغیر y بصورت زیر تعریف شده باشد:

Dim y as Variant

دستور زیر ایجاد خطا خواهد کرد:

Call Nstar(y)

در چنین مواقعی (وجود یک متغیر از نوع Variant) می توان آن را با استفاده از یک زوج پرانتز به نوع دلخواه تبدیل کرد:

Call Nstar ((y))

- مثال : تابعی بنویسید که دو متغیر x وy را که از نوع صحیح هستند، بعنوان آرگومان ورودی دریافت کرده و X^Y را بعنوان خروجی برگرداند (y مثبت فرض می شود).

```
Function Power (x as Integer، y as Integer) As Integer
   Dim p as Integer
   p=1
   For   I=1   to y
          p=p*x
   Next
   Power=p
End Function.
```

توجه : از نام تابع نمی توان در عملیات محاسباتی استفاده نمود، بلکه تنها می توان در آن مقداری جایگزین کرد.

برای صدا زدن تابع فوق می توان بصورت زیر عمل کرد:

```
Print Power (2,8)
A=power (5,2)
```

مثال: تابعی بنویسید که یک رشته از ورودی دریافت کند و معکوس آن را اعلام نماید.

```
Function Rev (Astr as String) as String
   R$=”“
   For I=1 to Len (Astr)
      C$ = Mid (Astr,I,1)
       R$=C$+R$
   Next
   Rev=R$
End Function.
```

- مثال : تابعی بنویسید که بزرگترین عنصر یک آرایه را بیابد.

```
Function MaxAr (P() As Integer، Last as Integer ) As Integer
   m% =P(1)
   For I=2    to Last
       If P(i)>m% then  m%=p(i)
   Next
   MaxAr =m%
End Function.
```

تابع فوق یک آرایه P باLast عضو را گرفته، بزرگترین عنصر آن را می یابد.

۵٤- متدهای Show و Hide

- برنامه هایی که تا اینجا نوشته ایم همه تنها دارای یک فرم بودند. برای اضافه کردن فرمهای دیگر کافی است از منوی Project گزینه Add Form را انتخاب نماییم. بعلاوه در پنجره Project Explorer می توانیم روی نام پروژه کلیک راست کرده، از منوی ظاهر شده گزینه Add و سپس From را انتخاب کنیم. (برای حذف فرم نیز از گزینه Remove استفاده می نماییم).

- بنابراین یک برنامه می تواند شامل چندین فرم باشد، یکی از فرم های برنامه، فرم اصلی می باشد که با اجرای برنامه بر روی تصویر ظاهر می شود. برای ظاهر یا مخفی کردن دیگر فرمها از دو متد Show و Hide استفاده می شود.

- مثال: می خواهیم بر روی فرم اصلی برنامه یک دکمه فرمان با نام و عنوان About قرار دهیم که با کلیک کردن بر روی آن فرم دومی، بنام فرم About که حاوی اطلاعاتی درباره برنامه مان است، ظاهر شود. بر روی فرم About دکمه فرمانی با نام و عنوان OK وجود دارد که لازم است که با کلیک کردن بر روی آن فرم About بسته شود.

برای ایجاد این برنامه پس از اضافه کردن فرم دوم به پروژه و قرار دادن دکمه های فرمان و سایر کنترل های لازم، کافیست در زیر برنامه های دکمه های فرمان، فرامین زیر را اضافه نماییم :

```
Sub About _ Click ()
    Form2 .Show
```

```
End Sub
Sub Oh – Click ()
      Form2.Hide
End Sub.
```

- توجه: استفاده از Hide تنها فرم را مخفی می کند، حال آنکه هنوز در حافظه است. برای آنکه حافظه از فرم خالی شود، از دستور Unload استفاده می کنیم :

```
Unload  Form2
```

۵۵- متد SetFocus

- این متد برای تمرکز روی یک کنترل بکار می رود. برای مثال اگر در یک فرم سه جعبه متن وجود دارد و می خواهیم با نمایش فرم، کرسر در جعبه متن سوم باشد، فرمان زیر را در رویداد Activate یا فعال شدن فرم اضافه می کنیم :

```
Sub Form1-Activate ()
      Text3-Setfocus
End Sub.
```

۵۶- اضافه کردن منوها

- می توان مانند سایر برنامه های محیط ویندوز، برای فرمها منو نیز تعبیه نمود. برای اینکار از منوی Tools گزینه Menu Editor را انتخاب می نماییم تا محیطی جهت اضافه کردن و مدیریت منوها در اختیارمان قرار گیرد.

ابزار قابل دسترس

Caption: عنوانی که در منو ظاهر می شود.

Name: نامی که به منو می دهیم و در کد برنامه از آن استفاده می کنیم.

Index: برای ایجاد آرایه ای از منوها. آرایه منوها مواقعی مفید است که بخواهیم در طول اجرا مواردی را به منو بیافزاییم.

WindowList : امکان ایجاد منویی که پنجره های باز را در برنامه های MDI (که چند پرونده را باهم باز میکنند) لیست کند.

Shortcut: میان بر یا کلیدهای تابع برای دسترسی به منو.

Checked: برای ایجاد منوی تیک خورده / نخورده.

Enabled: برای ایجاد منوی فعال / غیر فعال.

Visible: برای ایجاد منوی قابل رویت / غیر قابل رویت.

طرح کلی منو :

- دکمه های فلش دار: امکان تغییر سطح گزینه و تعیین محل گزینه منو.

می توان تا چهار سطح زیر منو برای هر منوی اصلی طراحی کرد.

- دکمه next:رفتن به سطر بعد(منوی بعد)یا اگر در انتهای منوها باشیم اضافه کردن یک گزینه منوی جدید.
- دکمه Insert:امکان اضافه کردن یک گزینه منوی جدید به بالای گزینه منویی که انتخاب شده (نوار رنگی بر روی آن قرار دارد).
- دکمهdelete:امکان حذف گزینه منوی انتخاب شده.

- **مثال:** برنامه ای بنویسید که حاوی سه منوی اصلی File، Color و Help باشد. در منوی File زیر منوی Save، Open و Exit را طراحی نمایید؛ بطوریکه با کلیک روی منوی Open، پنجره دیالوگ Open و با کلیک روی منوی Save، پنجره دیالوگSave ظاهر شود و با کلیک روی منویExit برنامه خاتمه یابد.

منوی Color شامل دو زیر منوی Blue و Red مورد نظر است که در ابتدای اجرای برنامه گزینهBlue تیک خورده و زمینه فرم نیزآبی می باشد. با انتخاب هر یک از دو منوی Blue و Red، گزینه مورد نظر تیک خورده و رنگ زمینه فرم به رنگ منوی مربوطه در می آید.

منوی Help تنها شامل یک زیر منوی About است که با کلیک بر روی آن فرم دومی درباره برنامه ظاهر میشود و با OK کردن فرم دوم، به فرم اول باز می گردیم.

منو را به صورت زیر در Form1 طراحی می کنیم :

File
….. Open
….. Save
….. Exit
Color
….. Blue
….. Red
Help
….. About

و نامهای زیر برای منوها در نظر می گیریم :

MnuFile, MnuOpen, MnuSave, MnuExit, MnuColor, MnuBlue, MnuRed, MneHelp, MnuAbout

بعلاوه برای منوی Blue، مورد Checked را انتخاب می نماییم.

در Form1، یک Command Dialog Box برای ظاهر کردن پنجره های Open و Save اضافه کرده، نام آن Cdb فرض می گنیم. با اضافه کردن Form2 در پروژه، اطلاعاتی را در مورد برنامه در Label ها می نویسیم و یک دکمه فرمان با عنوان و نام Ok در Form2 اضافه می نماییم.

کد زیر را در برنامه می نویسیم :

```
Sub MnuOpen _ Click ()
    Cdb.ShowOpen
End Sub

Sub MnuSave _ Click ()
    Cdb. ShowSave
End Sub.

Sub MnuExit _ Click ()
```

```
    End
End Sub

Sub MnuBlue _ Click ()
    Form1.BackColor = VbBlue
    MnuBlue.Checked = True
    MnuRed.Checked = False
End Sub

Sub MnuRed _ Click ()
    Form1. BackColor = VbRed
    MnuBlue.Checked = False
    MnuRed.Checked = True
End Sub

Sub MnuAbout _ Click ()
    Form2.Show
End Sub

Sub Ok _ Click ()
    Form2.Hide
End Sub.
```

۵۷- ایجاد فایل اجرایی برنامه

- برای ایجاد فایل اجرایی برنامه (با پسوند exe)، از منوی File گزینه Make Project1.exe را انتخاب می نماییم. (با فرض آنکه نام پروژه Project1 باشد.) در صورتیکه بخواهیم این برنامه را در کامپیوتری که ویژوال بیسیک روی آن نصب شده است اجرا کنیم مشکلی نداریم، زیرا فایلهای کتابخانه ای مانند فایلهایی با پسوند Dll در دایرکتوری System ویندوز موجودند. ولی برای اجرای این فایل اجرایی روی کامپیوترهایی که این نرم افزار را ندارند با مشکل روبرو خواهیم شد. چرا که باید بدانیم کدام فایلها مورد نیازند و باید آنها را به همراه فایل اجرایی جابجا نماییم.

- برای حل این مشکل با نصب ویژوال بیسیک ابزاری به نام Package And Deployment Wizard در اختیارمان قرار می گیرد که می توان برای ساختن فایل Setup.exe جهت نصب یک برنامه از آن استفاده کرد.

ابزاری برای تهیه یک بسته نرم افزاری

برای ساختن بسته های نرم افزاری (Software Package) معمول است که فایلهای مورد نیاز بصورت فشرده در فایلهای کابینت با پسوند Cab ذخیره شوند و با اجرای فایلSetup.exe توسط کاربر این فایلها از حالت فشرده خارج شده، هر یک در دایرکتوری مطلوب کپی شوند. ابزار Package and Deployment Wizard که به همراه ویژوال بیسیک نصب می شود، می تواند یک پروژه ویژوال بیسیک را کامپایل کرده، فایل اجرایی آن را بسازد. سپس فایلهای Dll لازم و سایر فایلهای مورد استفاده در پروژه (مانند تصاویر و غیره) را شناسایی نموده، آنها را در یک یا چند فایل کابینت فشرده کند. تعداد فایلهای کابینت به محدودیت حجم فایلها بستگی دارد که از کاربر سوال می شود. برای مثال اگر بخواهیم بسته نرم افزاری را بصورت مجموعه ای از دیسکت ها

ارایه دهیم، باید حجم فایلهای کابینت را به حجم دیسکت محدود کنیم. بعلاوه می توان در این ابزار مشخص کرد که برنامه پس از نصب به چه صورت در منوی Start نمایش داده شود. پس از تهیه فایلهای Setup.exe و فایلهای Cab آنها را به مشتری بسته نرم افزاری ارایه می کنیم. ارایه کد برنامه (فایلهای فرم و پروژه) اجباری نیست و بستگی به توافق بین شما و مشتری دارد. استفاده کننده از برنامه کافیست برنامه Setup.exe را اجرا کند تا برنامه مورد نظر روی کامپیوتر نصب شود. برای اجرا، از منوی Start میانبر انتخاب می شود.

۵۸- پرونده ها یا فایلها

- پرونده یا فایل: واحد ذخیره اطلاعات بر روی حافظه جانبی است. بعبارتی برای ذخیره هر گونه اطلاعات در کامپیوتر و حفظ آن حتی در صورت خاموش بودن کامپیوتر، لازم است آن را در یک فایل ذخیره کنیم.

- فایلها بر سه نوعند :
  - فایلهای متنی (Text)
  - فایلهای تصادفی (Random)
  - فایلهای باینری(Binary)

فایلهای متنی

این فایلها برای ذخیره و بازیابی متن استفاده میشود. دسترسی به اطلاعات این فایلها بصورت ترتیبی است. بعبارتی برای دسترسی به اطلاعاتی که در انتهای فایل وجود دارد، باید ابتدا از تمام اطلاعات قبل از آن روی فایل عبور کنیم تا به اطلاعات مورد نظر برسیم.

- **ایجاد وبازکردن یک فایل متنی**

فرض کنیم میخواهیم فایل test.txt را در درایو c ایجاد نماییم و روی آن بنویسیم. فرمان زیر لازم است :

Open “c:\test.txt“ for output as #1

یا شکل کلی دستور بصورت شماره فایل

Open نام خارجی فایل for نوع فایل as[#] شماره فایل

بعبارتی فایل c:\test.txt را در دستور فوق بعنوان فایل شماره یک و از نوع output باز کرده ایم.
نوعهای فایل در فایلهای متنی عبارتند از :

۱- output :در صورتیکه فایلی را از نوع out put باز کنیم، در صورتی که وجود نداشته باشد، ایجاد میشود و میتوان روی آن نوشت. در صورتی که وجود داشته باشد، محتویات قبلی آن پاک میشود و برای نوشتن مجدد آماده است.

۲- Append :در صورتیکه فایلی را از نوع Append باز کنیم، در صورتیکه وجود نداشته باشد، ایجاد میشود و میتوان روی آن نوشت. در صورتیکه وجود داشته باشد، محتویات قبلی حفظ شده و میتوان در ادامه فایل نوشت.

۳- Input :باز کردن فایل از نوع Input برای خواندن محتویات فایل مناسب است. در صورتیکه فایل موجود نباشد، اعلام خطا خواهد شد.

- استفاده از شماره های تکراری برای گشودن پرونده ها سبب بروز خطا خواهد شد.

- **بستن فایلها**

لازم است قبل از خارج شدن از برنامه، هر فایل باز را ببندیم. برای مثال، فایل شماره یک را که با دستور open باز کرده بودیم، بصورت زیر میبندیم.

```
Close # 1
```

استفاده از دستور close بدون ذکر شماره فایل، سبب بسته شدن کلیه فایلهای باز میشود.

- **نوشتن و خواندن بدون تشخیص انواع داده ای**

برای نوشتن یک سطر، در داخل یک فایل متنی، میتوان از دستور print به همراه شماره فایل استفاده نمود. برای مثال، دستورات زیر

```
Open “c:\test.txt“ for output  as #1
Print #1, “hello“
Print  #1, “how are you?“
Close #1
```

فایل c:\test.txt را باز کرده، دو سطر در آن مینویسد و سپس فایل را میبندد. میتوان این فایل در محیط یک ادیتور (مانند notepad) مشاهده کرد.

برای خواندن یک سطر از یک فایل متنی، میتوان از دستور Line Input به همراه شماره فایل استفاده کرد. برای مثال فرامین زیر :

```
Open  “c:\test.txt“ for input as #1
Line Input #1, a$
Line Input #1, b$
Close #1
```

فایل c:\test.txt را برای خواندن باز کرده، دو سطر از فایل را خوانده و سپس فایل را میبندد. پس از خواندن سطر اول در متغیر a$ و سطر دوم در متغیر b$ قرار خواهد گرفت. برای مثال، برای فایل test.txt که در مثال قبل نوشتیم، مقادیر دو متغیر b$ و a$ بصورت زیر خواهند بود :

```
a$= “ hello”
b$= “ how are you? “
```

- **نوشتن و خواندن با تشخیص انواع داده ای**

در دستوراتی که برای نوشتن و خواندن پیش از این بکار بردیم، نوشتن و خواندن یک سطر مطرح است و مشخص کردن انواع داده، مانند رشته ای یا عددی ممکن نمیباشد.

برای تشخیص نوع داده، هنگام نوشتن از دستور write و هنگام خواندن از دستور Input استفاده میکنیم. برای مثال:

```
open  "c:\test.txt" for output as #2
Write #2 "Ali", "Alavi", 18
Write #2, "Mahdi", "Mahdavi", 20
Close #2
```

در فرامین فوق، دو سطر توسط دستور write در فایل نوشته شده که هر سطر شامل دو رشته و یک عدد می باشد.

```
Open "c:\test2.txt" for input as #2
Input #2, Fname1$, Lname1$, age1%
Input #2, Fname2$, Lname2$, age2%
Close #2
```

دستور فوق، فایل را برای خواندن باز کرده دو سطر از فایل را می خواند. هر سطر شامل دو رشته و یک عدد صحیح است. پس از اجرای دستورات فوق، متغیرها دارای مقادیر زیر خواهند بود.

```
Fname1$ = " Ali "
Lname1$ = " Alavi "
Age1% = 18
Fname2$ = " Mahdi "
Lname2$ = " Mahdavi "
Age2% = 20
```

- **رسیدن به انتهای فایل**

برای اینکه بدانیم چه زمانی به انتهای فایل رسیده ایم و پس از آن اقدام به خواندن از فایل نکنیم، تابع Eof وجود دارد که رسیدن به انتهای فایل را اعلام میکند. برای مثال هر گاه به انتهای فایل شماره یک برسیم Eof (1) مقدار True خواهد داشت و در غیر این صورت مقدارش False است.

مثال : برنامه ای بنویسید که فایل readme.txt را خوانده، به صورت معکوس در فایل reverse.txt بنویسید.

```
Open "readme.txt" for input as #1
Open "reverse.txt" for output as #2
Do while not eof (1)
    Line input #1 , oneline$
    Print #2, reverse (oneline$)
Loop
Close

Function reverse (s$)
    T$ = " "
    For I =1 to len (s$)
        C$ = mid (s$, I,1)
        T$=c$+t$
    Next
```

```
    Reverese = t$
End function
```

فایلهای تصادفی(random)

- این نوع از پرونده ها برای ذخیره وبازیابی رکوردها مناسبند. یک رکورد برای ذخیره اطلاعات در مورد یک موجودیت بکار می رود. برای مثال برای موجودیت "دانشجو" می توان اطلاعاتی مانند شماره دانشجویی، نام، نام خانوادگی و سال تولد را در یک رکورد ذخیره کرد.
- برای تعریف رکوردی برای ذخیره اطلاعات یک دانشجو ابتدا نوع رکورد را تعریف کرده، سپس تغیری از ان نوع معرفی می کنیم :

```
Type  StudentRec
    Sno as string *10
    Fname as string*15
    Fname as string*30
    Bdate as integer
End type
Dim r  as StudentRec
```

در دستور فوق نوع رکوردی به نام StudentRec تعریف کرده ایم که شامل چهار قسمت sno (برای ذخیره یک شماره دانشجویی حداکثر ۱۰حرفی)، Fname (برای ذخیره نام دانشجو حد اکثر ۱۵ حرفی), Fname (برای ذخیره نام خانوادگی دانشجو با حد اکثر ۳۰ حرف) و Bdate (برای ذخیره سال تولد از نوع عدد صحیح) میباشد. سپس متغیری به نام r از این نوع رکورد تعریف کرده ایم که نتیجتاً r نیز شامل چهار قسمت فوق است. برای مثال برای دسترسی به این اجزا می توان بصورت زیر عمل نمود.

```
R.Sno =”80501234”
R.Fname=”ali”
R.Lname=”alavi”
R.age =18
```

- برای باز کردن یک فایل تصادفی که رکوردهایی از نوع مشخص را ذخیره نماید, به شکل کلی زیر عمل میکنیم :

Open نام پرونده for random as [#]شماره پرونده len = طول رکورد

برای مثال، برای باز کردن فایلی به نام test3.dat که رکوردهایی از نوع StudentRec را بتواند ذخیره کند, دستور زیر لازم است :

```
Open “test3.dat” for random as #1 len =len(r)
```

تابع len در دستور فوق طول رکوردr را محاسبه می کند. باز کردن فایل در مورد فایلهای تصادفی برای ایجاد, خواندن ونوشتن است و مانند فایلهای متنی، نیازی به محدودکردن مورد استفاده نداریم, زیرا دسترسی به فایلهای تصادفی از نوع تصادفی است. یعنی میتوان به هر رکورد از فایل بطور دلخواه دسترسی داشت, از آن خواند و یا بر روی آن نوشت.

- **مفهوم اشاره گر**

هنگامیکه در فایل ها می نویسیم یا از آنها می خوانیم, محل خواندن یا نوشتن توسط یک اشاره گر (pointer) مشخص می شود. هنگام باز کردن، فایل اشاره گر در ابتدای فایل قرار دارد, ولی میتوان آن را جابجا کرد. در فایلهای تصادفی، اشاره گر همواره به ابتدای یک رکورد از اطلاعات اشاره دارد. با خواندن یا نوشتن یک رکورد اشاره گر در ابتدای رکورد بعدی قرار خواهد گرفت. میتوان گفت اشاره گر در فایلهای تصادفی، همان شماره رکورد است.

- **خواندن و نوشتن در فایلهای تصادفی**

برای نوشتن در فایلهای تصادفی از دستور put وبرای خواندن از فایلهای تصادفی از دستور get با شکل کلی زیر استفاده میشود:

Put [#]شماره پرونده , [شماره رکورد], متغیر رکورد

Get [#] شماره پرونده, [شماره رکورد], متغیر رکورد

در صورتیکه شماره رکورد در دستورات فوق ذکر نشود, اطلاعات در محل قرار گیری اشاره گر خوانده یا نوشته میشود.

برای نوشتن اطلاعات یک دانشجو در فایل test3.dat در محل رکورد اول و خواندن اطلاعات از رکورد سوم میتوان به صورت زیر عمل نمود :

```
Type StudentRec
    Sno as string*10
    Fname as string*15
    Lname as string*30
    Age as integer
End type

Dim r as StudentRec
Open "test3.dat" for random as #1 len=len(r)
r.sno= "80501234"
r.fname= "ali"
r.lname= "alavi"
r.age=18
put #1,1,r
get #1,3,r
text1.text = r.Sno
text2.text = r.Fname
text3.txt = r.Lname
text4.txt = str (r.Age)
close #1
```

توجه کنید که برای خواندن از رکورد سوم, این رکورد باید قبلا نوشته شده باشد. در غیر اینصورت اعلام خطا خوهد شد. پس از خواندن از فایل اطلاعات دانشجوی سوم در چهار textbox نوشته میشود.

فایلهای باینری

فایلهای باینری شباهت زیادی به فایلهای تصادفی دارند, با این تفاوت که یک واحد اطلاعات در آنها به جای یک رکورد, یک بایت است. برای خواندن و نوشتن از دستورات put وget استفاده میشود و تنها باید محل نوشتن یا خواندن را بر حسب شماره بایت مشخص کرد.

مثال :

```
open “test4.bin” for binary as #1
A$ = “computer”
put # 1,1,A$
B$=”ing”
Put 1,7,B$
Close
```

پس از دستوزات فوق، محتوای پرونده test4.bin، Computing خواهد بود.

توجه: برای خواندن اطلاعات از فایلهای باینری، فقط میتوان از رشته های با طول ثابت استفاده کرد. برای مثال:

```
Dim Buffer as String*200
Get #1, 120, Buffer
```

دستورات فوق از بایت شماره ۱۲۰ شروع کرده، ۲۰۰ بایت را دز متغیر Buffer خواهند خواند.

دستور Seek

دستور Seek برای انتقال اشاره گر به موقعیت خاصی در فایل بکار میرود. شکل کلی آن بصورت زیر است:

Seek [#]شماره پرونده,شماره رکورد یا موقعیت

برای مثال، برای انتقال اشاره گر به رکورد دهم در فایل شماره یک، از نوع تصادفی:

```
Seek #1, 10
```

یا برای انتقال اشاره گر به بایت ۲۰۰ام در فایل شماره دو از نوع باینری:

```
Seek #2, 200
```