به نام خدا

جزوه

برنامه نویسی مقدماتی (ویژوال بیسیک)

رایانه کار پیشرفته

(استاندارد وزارت فرهنگ و ارشاد اسلامی)

تهیه و تنظیم:

هادی مومن

با همکاری خانم مهندس مریم شریفی

گروه کامپیوتر استان مرکزی(اراک)

زمستان ۹۰

واحد کار اول

توانایی حل مسایل و طراحی الگوریتم مناسب برای آنها

برای حل هر گونه مساله جدا از نوع آن باید موارد زیر را در نظر گرفت:

۱-شناخت دقیق مساله:

الف) ابتدا مقادیر معلوم مساله را مشخص می کنیم(داده ها).

ب) خواسته های مساله را مشخص می کنیم(مجهولات).

ج) ارتباط میان داده ها و مجهولات را پیدا کرده و در محاسبات حل مساله استفاده می کنیم.

۲-تجزیه و تحلیل مساله:

برخی از مسائل به سادگی قابل حل می باشند اما برای برخی مسائل جستجوی راه حل به سادگی امکان‌پذیر نمی باشد در این حالت مسئله را به مسائل (قسمت های) کوچکتر تقسیم می کنیم و این قسمت های کوچکتر را حل کرده و در نهایت با کنار هم قرار دادن آنها به حل کل مسئله می رسیم.

۳-طراحی راه حل مسئله:

پس از تحلیل مسئله می‌توانیم راه حل را ارائه کنیم که به دو روش امکان‌پذیر است:

الف) استفاده از تجربیات و راه حل های موجود

ب) استفاده از روش های تفکر منطقی و الگوریتمی که حل مسئله بر اساس آن صورت گرفته و به صورت مرحله به مرحله انجام می‌شود.

تعریف الگوریتم: به مجموعه‌ای از دستورالعمل ها که با زبان دقیق و قابل فهم به همراه جزئیات لازم و به صورت مرحله به مرحله به گونه‌ای اجرا شده که هدف خاصی (حل مسئله) را دنبال کنند و شروع و خاتمه آنها نیز مشخص باشد را الگوریتم می‌گویند.

شرایط الگوریتم:

۱ – استفاده از زبان ساده،دقیق و قابل فهم
۲ – استفاده از جزئیات کافی: یعنی دستورالعمل ها کامل اجرا شوند و به صورت کلی و مبهم نباشند.
۳ – الگوریتم و دستورالعمل های آن دارای شروع و پایان باشند.
۴ – دستورالعمل ها دارای ترتیب مشخصی باشند.
۵ – الگوریتم جامع باشد یعنی در تمام حالتها نتایج بدست آمده درست باشند (این نکته هم در نظر گرفت که بدون ایجاد خللی در جامع بودن آن کوتاه نیز باشد).

مثال: الگوریتمی که دو عدد را دریافت و با هم جمع کرده و حاصل را نمایش می دهد:

۱) شروع
۲) A و B را دریافت کن
۳) C ← A+B
۴) C را نمایش بده
۵) پایان

انواع دستورالعمل ها در الگوریتم:

۱ -**دستورالعمل های ورودی:** توسط این نوع دستورالعمل ها داده ها وارد الگوریتم می‌شوند که از عباراتی مثل " بگیر" ، " دریافت کن" یا "بخوان" استفاده می‌شود.

۲ -**دستورالعمل های خروجی:** توسط این نوع دستورالعمل ها نتایج الگوریتم نمایش داده می‌شوند و نیز برای نمایش پیام ها استفاده می‌گردند که در آنها از عباراتی مثل " نمایش بده" ، " چاپ کن" استفاده می‌شود.

۳ - **دستورالعمل های محاسباتی:** این دستورالعمل ها در واقع قسمت حل مسئله می‌باشند که به شکل کلی " عملیات محاسباتی = متغیر " می‌باشند . (بجای علامت تساوی از فلش نیز استفاده می شود)، دستورالعمل های مقداردهی نیز از این نوع هستند مثلا عبارت ۲ ⟵ x

تعریف متغیر: متغیر مکانی از حافظه است که برای نگهداری موقت داده ها و اطلاعات استفاده می شود و در طول اجرای الگوریتم(برنامه) مقدار آن تغییر می‌کند.

تعریف عملگر: یک یا مجموعه عملیاتی است که از قبل در الگوریتم آماده شده(تعریف شده) و برای استفاده از آنها علایمی به صورت قراردادی تعریف شده‌اند مثل عملگر جمع (+) .

عملگرهای ریاضی :

به ترتیب تقدم:

( )

^ توان

* / ❋

\ (خارج قسمت) تقسیم صحیح

mod باقیمانده

+ - (اگر دو یا چند عملگر تقدم یکسانی داشتند ابتدا عملگری که سمت چپ‌تر می‌باشد انجام می‌شود)

مثال : 2 * 4 + (6 - 2) ^ 2 / 4 - (6 \ 4) نتیجه عبارت بعد از اعمال تقدم عملگرها: 11

۴ ۶ ۱ ۳ ۵ ۷ ۲

مثال: الگوریتمی که شعاع یک دایره را دریافت و محیط و مساحت آنرا محاسبه و نمایش می دهد:

۱) شروع
۲) R را دریافت کن
۳) P ⟵ 2 * 3.14 * R
۴) S ⟵ 3.14 * R ^ 2
۵) P و S را نمایش بده
۶) پایان

٤ -**دستورالعمل های شرطی:** گاهی نیاز است روند اجرای دستورات با اتخاذ تصمیمات مناسب و بر اساس شرط یا شرطهایی انجام پذیرد در اینصورت از این نوع دستورالعمل ها استفاده می‌شود.

شکل کلی :

اگر شرط(ها) آنگاه دستور(ات)

اگر شرط(ها) آنگاه دستور(ات) ۱ در غیر اینصورت دستور(ات)۲

مثال: الگوریتمی بنویسید که یک عدد صحیح دلخواه را دریافت و با پیغام مناسب مشخص کند آن عدد زوج است یا فرد.

۱) شروع
۲) X را دریافت کن
۳) اگر X mod 2 = 0 آنگاه چاپ کن " زوج است " در غیر اینصورت چاپ کن " فرد است"
۴) پایان

عملگرهای مقایسه‌ای :

از این عملگرها در عبارات شرطی استفاده می شود.

کوچکتر	<
بزرگتر	>
کوچکترمساوی	< =
بزرگترمساوی	> =
مساوی	=
نامساوی	< >

(تقدم همه یکسان است)

مثال: الگوریتمی بنویسید که سه عدد دلخواه را دریافت و بزرگترین آنها را تعیین و نمایش دهد.

۱) شروع
۲) A ، B و C را دریافت کن
۳) Max ← A
۴) اگر B > Max آنگاه Max ← B
۵) اگر C > Max آنگاه Max ← C
۶) Max را چاپ کن
۷) پایان

عملگرهای منطقی:

ازاین عملگر ها برای ترکیب شرط های مختلف استفاده می شود.(در برخی الگوریتم ها در دستورالعمل های شرطی لازم است بیش از یک شرط مورد بررسی قرار گیرد و با توجه به نتیجه تمام شرطها دستورالعملها اجرا شوند که در چنین حالتهایی از عملگرهای منطقی استفاده می‌شود)

عملگر **And** ("و" منطقی):

اگر چند شرط با And ترکیب شوند در صورتی نتیجه کل عبارت Trueمی شود که همه شرط ها Trueباشند.

A	B	A and B
F	F	F
F	T	F
T	F	F
T	T	T

عملگر **OR** ("یا" منطقی):

اگر چند شرط با Or ترکیب شوند در صورتی نتیجه کل عبارت Trueمی شود که حداقل یکی از شرط ها Trueباشد.

A	B	A and B
F	F	F
F	T	T
T	F	T
T	T	T

عملگر **Not** (نقیض):

عملگر Not ارزش یک شرط یا ترکیب شرط‌ها را معکوس می کند.

A	Not (A)
F	T
T	F

اولویت اجرای (تقدم) عملگرهای منطقی به ترتیب Not و And و سپس Or می باشد.

۵ -دستورالعمل های تکرار(حلقه ها):

در برخی الگوریتم ها لازم است برخی دستورالعمل ها به دفعات تکرار شوند در اینصورت از این نوع دستورالعمل ها استفاده می شود.

اجزای دستورالعمل های تکرار(حلقه ها):

الف) **شمارنده حلقه:** متغیری عددی است که تعداد دفعات تکرار دستورالعمل ها را در حلقه کنترل می کند.

ب) **مقدار اولیه:** مقدار اولیه را برای شمارنده حلقه تعیین می کند و قبل از شروع حلقه تعیین می شود.

ج) **شرط حلقه:** این شرط پایان دستورات حلقه را تعیین می کند و در واقع برای کنترل تعداد دفعات تکرار دستورات حلقه استفاده می شود.

د) **دستورات حلقه:** دستورالعمل هایی هستند که داخل حلقه تکرار می شوند.

مثال هایی از الگوریتم ها:

الگوریتمی بنویسید که محتویات دو متغیر را با هم عوض کند.

۱) شروع
۲) A و B را بگیر
۳) $C \leftarrow A$
۴) $A \leftarrow B$
۵) $B \leftarrow C$
۶) A و B را چاپ کن
۷) پایان

الگوریتم قبل را بدون استفاده از متغیر سوم بنویسید.

۱) شروع
۲) A و B را بگیر
۳) $A \leftarrow A + B$
۴) $B \leftarrow A - B$
۵) $A \leftarrow A - B$
۶) A و B را چاپ کن
۷) پایان

الگوریتمی بنویسید که اعداد طبیعی کمتر از ۱۰۰ را چاپ کند.

۱) شروع
۲) $X \leftarrow 1$
۳) X را چاپ کن
۴) $X \leftarrow X + 1$
۵) اگر $X<100$ آنگاه برو به مرحله ۳
۶) پایان

الگوریتمی قبل را برای اعداد طبیعی کمتر از N بنویسید.

۱) شروع
۲) N را دریافت کن
۳) $X \leftarrow 1$
۴) X را چاپ کن
۵) $X \leftarrow X + 1$
۶) اگر $X< N$ آنگاه برو به مرحله ۴
۷) پایان

الگوریتمی بنویسید که مجموع اعداد طبیعی زوج کمتر از N را محاسبه و چاپ کند.

۱) شروع
۲) N را دریافت کن
۳) $Sum \leftarrow 0$
۴) $X \leftarrow 2$
۵) $Sum \leftarrow Sum + X$
۶) $X \leftarrow X + 2$
۷) اگر $X< N$ آنگاه برو به مرحله ۵
۸) Sum را چاپ کن
۹) پایان

الگوریتمی بنویسید که N فاکتوریل را محاسبه و چاپ کند. (!۶ یعنی $۶ \times ۵ \times ۴ \times ۳ \times ۲ \times ۱$)

۱) شروع
۲) N را دریافت کن
۳) $F \leftarrow 1$

۴) X ← 1
۵) F ← F * X
۶) X ← X + 1
۷) اگر X<=N آنگاه برو به مرحله ۵
۸) F را چاپ کن
۹) پایان

الگوریتمی بنویسید که مجموع ارقام یک عدد طبیعی دلخواه را محاسبه و چاپ کند.

۱) شروع
۲) N را دریافت کن
۳) Sum ← 0
۴) d ← N mod 10
۵) Sum ← Sum + d
۶) N ← N \ 10
۷) اگر N< > 0 آنگاه برو به مرحله ۴
۸) Sum را چاپ کن
۹) پایان

الگوریتمی بنویسید که نمرات ۱۰ درس یک دانش آموز را دریافت و میانگین (معدل) او را محاسبه و چاپ کند.

۱) شروع
۲) X ← 1
۳) Sum ← 0
۴) Nom را دریافت کن
۵) Sum ← Sum + Nom
۶) اگر X < = 10 آنگاه برو به مرحله ۴
۷) Avg ← Sum / 10
۸) Avg را چاپ کن
۹) پایان

واحد کار دوم

توانایی ترسیم فلوچارت

در الگوریتم های کوچک فهمیدن الگوریتم و تبدیل آن ها به برنامه به سادگی انجام می‌گیرد اما در الگوریتم های بزرگ و پیچیده این کار دشوار است بنابراین باید از روش‌های مفیدتری استفاده کرد که فلوچارت یکی از آنهاست که در آن از اشکال و ترسیمات قراردادی برای نمایش دستورالعمل ها استفاده می‌شود که باعث درک بهتری از نحوه اجرای دستورات می گردد.

علایم و اشکال در فلوچارت

۱) **علایم شروع و پایان :** از بیضی استفاده می‌شود

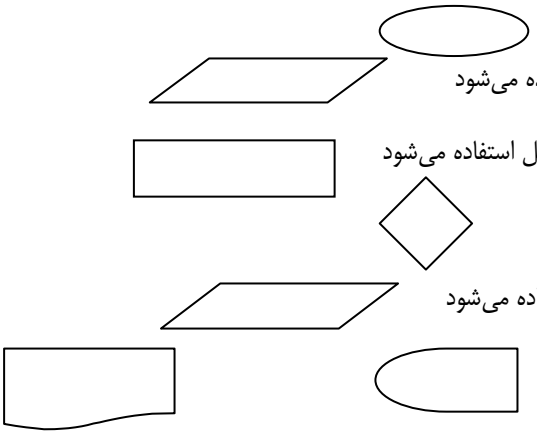
۲) **علایم دستورات ورودی:** از متوازی‌الاضلاع استفاده می‌شود

۳) **علایم دستورات انتساب(محاسباتی) :** از مستطیل استفاده می‌شود

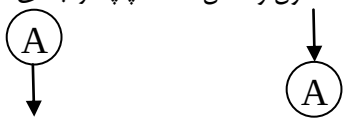
٤) **علایم دستورات شرطی:** از لوزی استفاده می‌شود

۵) **علایم دستورات خروجی:** از متوازی‌الاضلاع استفاده می‌شود

از اشکال مقابل نیز استفاده می‌شود

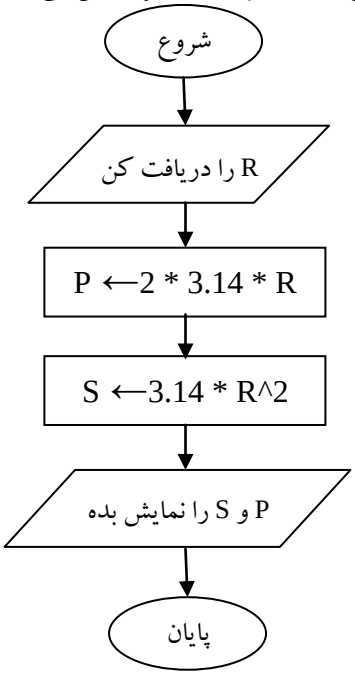


٦) **علامت اتصال :** در مواقعی که فلوچارت به علت بزرگی در یک صفحه(کاغذ) قرار نمی‌گیرد در این صورت از علامت زیر برای اتصال استفاده می‌شود (شکل سمت راست در انتهای فلوچارت در صفحه اول و شکل سمت چپ در ابتدای فلوچارت(ادامه آن) در صفحه دومی قرار می‌گیرد).

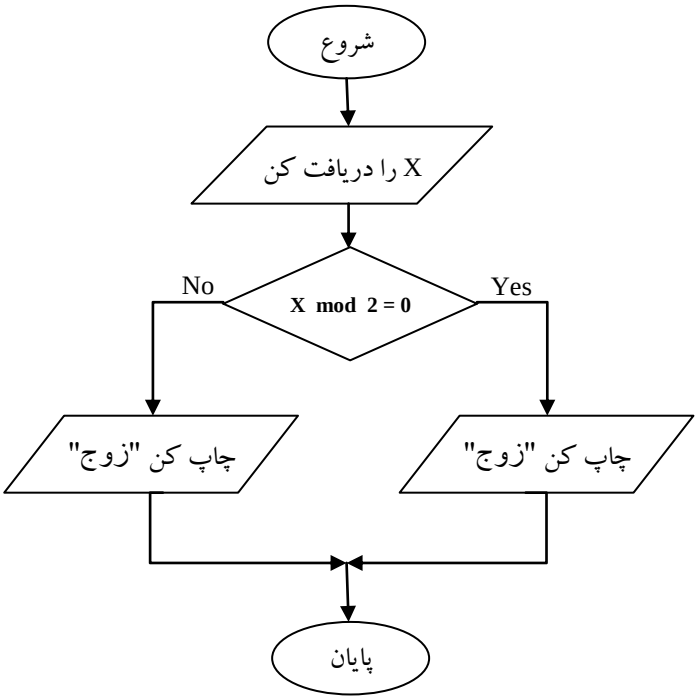


مثال هایی از فلوچارت الگوریتم های واحد کار اول :

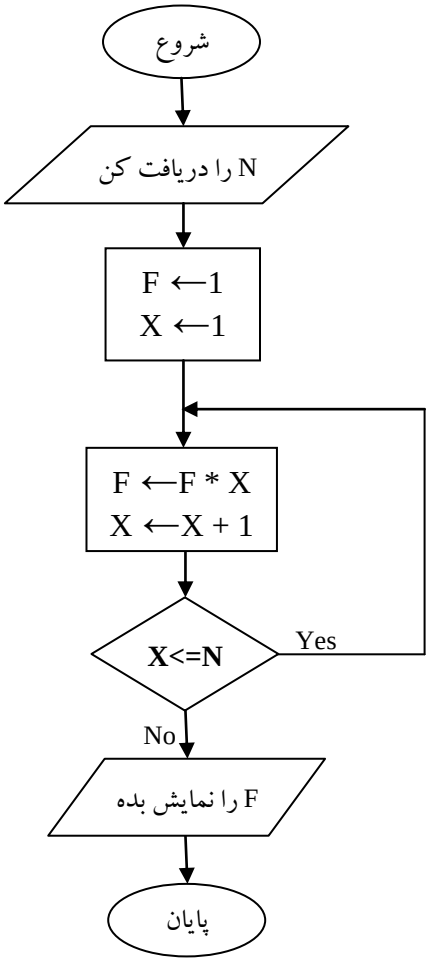
مثال: فلوچارتی که شعاع یک دایره را دریافت و محیط و مساحت آنرا محاسبه و نمایش می دهد:



مثال: فلوچارتی که یک عدد صحیح دلخواه را دریافت و با پیغام مناسب مشخص می کند آن عدد زوج است یا فرد.



مثال: فلوچارتی که N فاکتوریل را محاسبه و چاپ می کند. (!۶ یعنی ۶ × ۵ × ۴ × ۳ × ۲ × ۱)



نکته: تعداد فلش‌هایی که می‌تواند به هر علامت وارد شود نامحدود است اما در تمام علایم بجز علامت شرط فقط یک فلش می‌تواند خارج شود. (توضیح اینکه از علامت شرط باید فقط ۲ فلش خارج شود).

واحد کار سوم

توانایی درک و شناخت زبان برنامه نویسی ویژوال بیسیک و ایجاد یک برنامه کاربردی

نرم افزارها مجموعه‌ای از داده‌ها و دستورالعملها هستند که بوسیله برنامه‌نویس و بر اساس قواعد مشخص نوشته می‌شوند و سخت افزار را قابل استفاده می‌کنند.

انواع نرم‌افزار :

۱) سیستمی :
الف) سیستم های عامل
ب) زبان های برنامه نویسی
ج) برنامه های سودمند

۲) کاربردی

به مجموعه‌ای از قواعد و دستورالعملهای تعریف شده **زبان برنامه‌نویسی** می‌گویند.

انواع زبان برنامه‌نویسی:

۱) **زبان های سطح پایین :** ماشین (صفر و یک) ، اسمبلی
۲) **زبان های سطح میانی : C و C++**
۳) **زبان های سطح بالا : فرترن ، بیسیک ، پاسکال**

از زبان **فرترن** در برنامه‌نویسی علمی که نیاز به دقت بالا در محاسبات است استفاده می‌شود.

از زبان **پاسکال** که با ویژگی ساخت یافته است در برنامه‌نویسی علمی و تجاری استفاده می‌شود.

از زبان **C** که با ویژگی ساخت یافته است و زبان برنامه‌نویسی سیستم است برای طراحی و تولید هر نوع نرم‌افزاری می‌توان استفاده کرد.

زبان برنامه نویسی **ویژوال بیسیک** حاصل توسعه و ارتقای زبان بیسیک است که یک زبان ساخت یافته و شیء گرا می باشد.

ویژگی های زبان برنامه نویسی شیءگرا (ویژوال بیسیک):
ویژگی GUI (رابطه گرافیکی کاربر) : توسط این ویژگی می توان به آسانی برنامه های تحت ویندوز ایجاد کرد و کاربر یا برنامه‌نویس با محیط برنامه بطور ساده رابطه برقرار می کند.
ویژگی RAD (طراحی سریع برنامه) : توسط این ویژگی می توان در VB برنامه های کاربردی را با استفاده از ویزارد ها و ابزار ها به سرعت طراحی و تولید و گسترش داد.
ویژگی Event Handling (رویداد گرا) : این ویژگی خاصیت رویداد گرایی و مدیریت بر اساس اتفاقات را نشان می دهد.
ویژگی Error Handling (خطا یابی): VB امکانات بسیار مناسبی در زمینه کشف خطاهای نوشتاری و منطقی به برنامه نویس ارائه می دهد و نیز این امکان را می دهد خطاهای غیر قابل پیش بینی در هنگام اجرا تشخیص داده و راه حل مناسب به کاربر ارائه می دهد.
ویژگی IDE (محیط توسعه یافته مجتمع) : این ویژگی به برنامه نویس اجازه می‌دهد تا برنامه های خود را به سهولت و سرعت تحت ویندوز بدون نیاز به استفاده از برنامه های کاربردی دیگر طراحی، ایجاد، خطا یابی و اجرا کند.
API : بوسیله توابع Api می توان به امکانات و برنامه های کاربردی و داخلی ویندوز دسترسی پیدا کرد.
ویژگی OOP (شیء گرایی) : این ویژگی نشان دهنده شیء گرا بودن محیط VB می باشد.

با اجرای برنامه VB پنجره ای به نام New project باز می شود که دارای سه زبانه است :

زبانه **New** : برای ایجاد پروژه و برنامه جدید استفاده می شود (Standard EXE این نوع برنامه ای با فایل های اجرایی مستقل برای اجرا در محیط ویندوز طراحی می کند).

زبانه **Existing** : توسط این زبانه می توان برنامه های ذخیره شده در حافظه ها را باز کرد.

زبانه **Recent** : این زبانه لیستی از پروژه ها و برنامه هایی که اخیراً ایجاد و یا استفاده شده اند را برای دسترسی سریع به آنها در اختیار شما قرار می دهد.

معرفی پنجره ها و اجزای محیط برنامه VB

Form (پنجره طراحی فرم) : به وسیله این پنجره می توان تغییرات لازم را روی فرم که محل طراحی برنامه می باشد اعمال کرد (اگر بسته باشد بوسیله دکمه View object در پنجره پروژه ویا گزینه object از منوی view و یا کلید میانبر Shift + F7 می توان آنرا باز کرد).

ToolBox (جعبه ابزار): این کادر شامل یا حاوی آیکن کنترل های مختلف برای ایجاد اشیاء می باشد.

Project Explorer (پنجره پروژه) : بوسیله این پنجره به تمام فایل ها و اجزای تشکیل دهنده برنامه می توان دسترسی داشت (می توان ویرایش، ذخیره سازی یا حذف اجزا را درآن انجام داد). با کلید میانبر Ctrl+R می توان آنرا باز کرد.

Properties Window (پنجره خواص یا ویژگی ها): بوسیله آن می توان خواص و ویژگی های مربوط به فرم ها و کنترل ها (اشیاء) را مشاهده و تنظیم کرد. با کلید F4 می توان آنرا باز کرد.

Form Layout Window (پنجره تعیین موقعیت) : این پنجره برای تعیین موقعیت فرم (محل قرار گیری فرم) روی صفحه نمایش (دسکتاپ) در هنگام اجرای برنامه استفاده می شود.

معرفی چند کنترل

کنترل **Label** (بر چسب) : از این کنترل برای قرار دادن متن و پیغام روی فرم استفاده می شود.

کنترل **TextBox** (کادر متن) : از این کنترل برای وارد کردن داده ها به برنامه استفاده می شود. (داده ها را از نوع رشته ای دریافت می کند)

کنترل **CommandButton** (دکمه فرمان): از این کنترل برای ایجاد دکمه فرمان برای اجرای برخی برنامه ها و دستورات استفاده می شود ، (با کلیک روی آن یا زدن کلید Enter وقتی فوکوس روی آن است).

روش اجرای برنامه و گزینه های Run

۱- زدن دکمه Start در نوار ابزار استاندارد

۲- گزینه Start از منوی Run

۳- کلید میانبر F5

(از دستور Break ویا کلید میانبر Ctrl + Break برای متوقف کردن اجرای برنامه (استراحت و توقف) و از دستور End برای پایان اجرای برنامه استفاده می شود. (اگر اجرای برنامه را متوقف کنیم می توانیم با زدن فرمان Continue از منوی Run اجرای برنامه را ادامه دهیم یا با دستور Restart از ابتدا اجرا کنیم)).

برای **ذخیره برنامه** یا پروژه از منوی فایل گزینه Save Project را می زنیم پنجره ای باز می شود ابتدا اجزای برنامه (مثلاً فرم ها) را ذخیره و سپس خود پروژه را ذخیره می کنیم.

نکته: پسوند فایل های فرم در ویژوال بیسیک frm می باشد.

پسوند فایل های پروژه در ویژوال بیسیک vbp می باشد.

برای **اضافه کردن فرم جدید** به برنامه از یکی از روش های زیر استفاده می شود:

۱. در پنجره پروژه راست کلیک کرده گزینه Add و سپس Form را می زنیم و از پنجره باز شده گزینه Form و سپس دکمه Open را می زنیم.
۲. از نوار استاندارد دکمه Add form → Form →Open
۳. از منوی Project گزینه Add form → Open

برای تعیین اینکه کدام فرم ابتدا اجرا شود:

از منوی Project گزینه Project properties… و از پنجره باز شده از زبانه General از قسمت Startup Object فرم مورد نظر را انتخاب می‌کنیم.

معرفی چند ویژگی(خاصیت) اشیاء و فرم

خاصیت **Name** : هر شیء یا فرم در محیط ویژوال بیسیک بخصوص درمحیط کد نویسی (ماژول کد) با این خاصیت شناخته می شود.

خاصیت **Caption** : از این خاصیت برای تعیین **عنوان** اشیاء و فرم ها استفاده می شود.

خاصیت **Height** : از این خاصیت برای تنظیم اندازه **ارتفاع** اشیاء و فرم ها استفاده می شود.

خاصیت **Width** : از این خاصیت برای تنظیم اندازه **عرض(پهنا)** اشیاء و فرم ها استفاده می شود.

خاصیت **Left** : از این خاصیت برای تنظیم فاصله اشیاء از سمت چپ فرم و در مورد فرم ها از سمت چپ صفحه نمایش(دسکتاپ) استفاده می شود.

خاصیت **Top** : از این خاصیت برای تنظیم فاصله اشیاء از بالای فرم و در مورد فرم ها از بالای صفحه نمایش(دسکتاپ) استفاده می شود.

نکته : واحد اندازه گیری طول در VB بطور پیش فرض **twip** می باشد (هر twip برابر $\frac{1}{567}$ سانتی متر می باشد).

خاصیت **Alignment** : از این خاصیت برای تراز بندی متن اشیاء استفاده می شود.

خاصیت **Backcolor** : از این خاصیت برای تنظیم رنگ پس زمینه اشیاء و فرم ها استفاده می شود.

خاصیت **Forecolor** : از این خاصیت برای تنظیم رنگ قلم (متن یا عبارت) اشیاء و فرم ها استفاده می شود(رنگ پیش زمینه).

نحوه قرار دادن کنترل ها به عنوان شیء روی فرم

به دو روش می توان این کار را انجام داد:

۱- با دابل کلیک روی کنترل موردنظر درToolbox که با اندازه ثابتی در وسط فرم شیء مربوطه ایجاد می شود.

۲- با کلیک کردن روی کنترل مورد نظر درToolbox آنرا انتخاب و با اندازه دلخواه و در موقعیت مورد نظر با درگ کردن شیء مربوطه ایجاد می شود.

مفهوم **رویداد:** به هر یک از اتفاقاتی که در محیط ویندوز رخ می دهد یک رویداد(Event) می گویند مثل بستن و باز کردن پنجره‌ها ، کلیک ، دابل کلیک ، فشار دادن دکمه های موس یا کلیدهای صفحه کلید و ...

شکل کلی یک برنامه(رویه) رویدادی:

Private Sub Form یا کلمه (شیء) کنترل نام _ نام رویداد ( )

دستورات

End Sub

توضیح اینکه وقتی رویداد روی شیء یا فرم اتفاق می‌افتد دستورات اجرا می‌شوند

معرفی چند رویداد (Event)

رویداد **Click** : این رویداد هنگامی که روی اشیاء یا فرم کلیک می کنیم اتفاق می افتد.

رویداد **Dblclick** : این رویداد هنگامی که روی اشیاء یا فرم دابل کلیک می کنیم اتفاق می افتد.

رویداد **Load** : این رویداد مخصوص فرم می باشد و زمانی که برنامه را اجرا می کنیم روی فرم اتفاق می افتد (در واقع زمانی که اطلاعات فرم وارد حافظه شده و فرم بارگذاری می شود اتفاق می افتد).

رویداد **Change** : این رویداد مختص Textbox میباشد و هنگامی رخ می دهد که تغییری در آن(کادر متن) ایجاد شود.

از دو روش برای **تغییر خصوصیات** اشیاء و فرم ها استفاده می شود:

۱ - استفاده از پنجره خواص (Properties)

۲ - استفاده از کد نویسی: **مقدار خاصیت = نام خاصیت . نام شیءیا فرم**

مثال : Form1. BackColor = VBBlue

Label1 . Caption = "Valiasr"

نکته: عبارات متنی(رشته ای) را در مقداردهی داخل دابل کوتیشن " " قرار می دهیم.

معرفی چند ویژگی(خاصیت) کنترل کادر متن

خاصیت **Text** : این خاصیت مختص کنترل TextBox می باشد وداده یا عبارت داخل آن در این خاصیت قرار می گیرد. (Text1.text)

خاصیت **ScrollBars** : بوسیله این خاصیت می توانید انواع متفاوتی از نوارهای پیمایش را برای کنترل TextBox (کادر متن) قرار دهید. برای فعال شدن این خاصیت (نوار های پیمایش) ابتدا باید مقدار خاصیت MultiLine را برابر True کرد.

خاصیت **MultiLine** : از این خاصیت برای چند خطی کردن کنترل کادر متن(Text Box) استفاده می شود (یعنی اگر مقدار این خاصیت True شود در کنترل کادر متن می توان با زدن کلیدEnter به خط بعدی رفت و آنجا تایپ کرد).

واحد کار چهارم

توانایی تعریف انواع متغیرها ، ثابت ها و استفاده از عملگرهای ریاضی و رشته‌ای

متغیر

تعریف متغیر : مکانی از حافظه است که برای ذخیره داده ها و اطلاعات بطور موقت در برنامه ها استفاده می شود و در طول اجرای برنامه مقدار آن تغییر می کند.

روش تعریف متغیر: **Dim نام متغیر as نوع داده**

قوانین نامگذاری متغیر ها: (برای نامگذاری متغیرها از کاراکتر های حرفی، رقمی یا ترکیبی استفاده می شود)

۱. حد اکثر طول متغیر ۲۵۵ کاراکتر و حداقل یک کاراکتر باید باشد.
۲. حتماً باید با یک کاراکتر حرفی شروع شود.
۳. استفاده از علائم نقطه (.)، کاما (,)، تفریق (-) و فاصله خالی مجاز نمی باشد (اما استفاده از کاراکتر زیر خط مجاز(_) است).
۴. در نامگذاری از اسامی یا علائم رزرو شده استفاده نمی شود (یعنی چیزهایی که برای VB معنی خاصی دارند) مثلاً form , control , - , + و . . .
۵. VB بین کاراکتر های حرفی کوچک و بزرگ تفاوتی قایل نمی شود. (مثلاً a1, A1 با هم تفاوتی ندارند).

انواع داده ها:

فضای اشغال حافظه	محدوده مقادیر (توضیح)	نوع داده	
۱ بایت	۰ تا ۲۵۵	Byte	صحیح
۲بایت	۳۲۷۶۸- تا ۳۲۷۶۷	Integer	
۴ بایت	صحیح بلند(طولانی) ۹ رقمی	Long	
۴ بایت	اعشاری معمولی	Single	اعشاری
۸ بایت	اعشاری مضاعف	Double	
۸ بایت	مبالغ ارزی(پولی)	Currency	
۲ بایت	True یا False	Boolean	منطقی
۸ بایت	ساعت و تاریخ	Date	
برای هر کاراکتر یک بایت	* رشته ای با طول ثابت از ۱ تا ۶۵۴۰۰ کاراکتر * رشته ای با طول متغیر از صفر تا ۲ میلیارد کاراکتر(۲×۱۰^{۹})	String	
۱۶ بایت در مورد داده های عددی و داده های دیگر با توجه به نوع داده متغیر	هر نوع داده ای	Variant	

متغیر های رشته ای:

اگر بخواهیم مقادیر غیر عددی مانند متون وکاراکتر ها را ذخیره کنیم از این نوع داده استفاده می کنیم که به دو شکل تعریف می شود:

۱ – متغیر رشته ای با طول ثابت :

Dim نام متغیر as String*n

n طول رشته (طول متغیر) می باشد در واقع n حد اکثر تعداد کاراکتر ها را مشخص می کند (n بایت از حافظه اشغال می شود)

مثال: Dim name as String*20 ۲۰ بایت از حافظه

۲ – متغیر رشته ای با طول متغیر:

Dim نام متغیر as String

به اندازه رشته ذخیره شده در آن حافظه اشغال می کند.

مثال: Dim name as String

Name = "Ali" سه بایت از حافظه

نکته: اگر به متغیری نیاز داشتیم که نوع داده هایی که در آن ذخیره می کنیم مشخص نبود و بخواهیم انواع داده های مختلفی در آن ذخیره کنیم از نوع داده ی Variant استفاده می کنیم (به عبارتی به جای همه انواع داده ها می توان از آن استفاده کرد).

نکته: در صورتی که هنگام تعریف متغیر ، نوع آن تعیین نشود به طور خودکار از نوع Variant در نظر گرفته می شود:

Dim x X از نوع Variant

متغیر ضمنی : علاوه بر روش قبلی (دستور Dim) به روش ضمنی نیز می توان متغیرها را تعریف کرد

(1) Dim x as Integer
x=3

دستورات ۱ و ۲ معادل هستند

(2) x%=3

نوع داده	کاراکتر ضمنی
Integer	%
Long	&
Single	!
Double	#
Currency	@
String	$

مقدار دهی به متغیرها: برای مقداردهی به متغیرها از دستور Let استفاده می کنیم ، مثلا:

صحیح Let x = 10

اعشاری Let A1 = 2.5

منطقی (Boolean) Let b1 = True (بجای True از صفر و بجای False از یک نیز می توان استفاده کرد)

مقادیر رشته ای را داخل دابل کوتیشن قرار می دهیم Let name = "Ali"

مقادیر از نوع تاریخ و ساعت (Date) را داخل علامت نامبر (#) قرار می دهیم Let t1 = #2012/1/10 10:02:30 Am#

نکته: استفاده از دستور Let اختیاری است یعنی می توانیم آنرا ننویسیم.

نکته: اگر متغیری را تعریف کنیم و به آن مقدار ندهیم مقادیر پیش فرض آن بصورت زیر است:

نوع داده	مقدار پیش فرض
صحیح (Byte , Integer , Long)	0
اعشاری (Single , Double , Currency)	0.0
منطقی (Boolean)	False
تاریخ و ساعت (Date)	00:00:00
رشته ای (String)	رشته تهی ("")

دستور **Unload :** از این دستور برای بستن فرم استفاده می شود (فرم را از حالت لود (اجرا) خارج می کند)

Unload form1
Unload Me بجای نام فرم در ماژول کد آن فرم می توان از کلمه me استفاده کرد

متد : یک یا مجموعه عملیاتی است که روی شی ء یا فرم انجام می شود و روی آن تاثیر می گذارد.

شکل کلی: **متد . نام شی ء یا فرم**

Form1.cls

متد **Setfocus :** با اجرای این متد فوکوس روی شی یا فرم موردنظر قرار می گیرد.

(فوکوس امکانی است که با استفاده از موس یا کیبورد به کاربر اجازه می‌دهد به شیء مورد نظر دسترسی پیدا کند یعنی آنرا فعال یا انتخاب کند.)

مثال: Text1.setfocus

✍ در پنجره کدنویسی **General** بخش **تعاریف عمومی** برنامه می باشد و در آن برخی دستورات خاص VB بصورت عمومی قرار می‌گیرد.

نکته: اگر متغیری از قبل تعریف نشود ولی استفاده شود در هنگام اجرای برنامه پیام خطایی نمایش داده می‌شود و اجرای برنامه متوقف می‌شود (Variable not defined یعنی متغیر تعریف نشده است) و آن قسمت از برنامه به رنگ آبی می‌شود. (در واقع این خطا هنگامی دیده می‌شود که در بخش تعاریف عمومی دستور Option Explicit نوشته شده باشد یا از منوی Tools گزینه Option و از پنجره باز شده گزینه Require Variable Declaration انتخاب شده باشد و در غیر اینصورت پیام خطایی رخ نمی‌دهد و متغیر بطور خودکار از نوع Variant تعریف می‌شود).

✍ **تابع Val**: این تابع مقدار نوع **رشته‌ای** را به نوع داده **عددی** تبدیل می‌کند.

مثال: Val(Text1.text)

✍ **عملگرهای ریاضی (محاسباتی):**

به ترتیب تقدم:

( )

^ توان

* /

\ (خارج قسمت) تقسیم صحیح

mod باقیمانده

+ - (اگر دو یا چند عملگر تقدم یکسانی داشتند ابتدا عملگری که سمت چپ‌تر می‌باشد انجام می‌شود)

مثال : 2 * 4 + (6 - 2) ^ 2 / 4 - (6 \ 4) نتیجه عبارت بعد از اعمال تقدم عملگرها: 11

(۴) (۶) (۱) (۳) (۵) (۷) (۲)

✍ **ثابت:** ثابت نیز همانند متغیر می‌باشد باید ابتدای برنامه تعریف شود و فقط تفاوت آن این است که مقدار آن در طول اجرا تغییر نمی‌کند.

شکل کلی: **Const نام ثابت as نوع داده = مقدار**

مثال: Const Pi as single = 3.14

✍ **بسته به نوع تعریف متغیر،۳ نوع متغیر وجود دارد :**

۱- **متغیر محلی:** اگر یک متغیردر یک رویه رویدادی تعریف شود آن متغیر فقط در آن رویه ودر دستورات آن قابل دسترسی وشناسایی می باشد به این نوع متغیر محلی(private)می گویند.

۲- **متغیر عمومی:** گاهی اوقات لازم است یک متغیر در تمام رویه ها شناخته شود واز آن استفاده گردد برای این کار آن را دربخش تعاریف عمومی (General)تعریف می کنیم (برای تعریف متغیر عمومی به جای کلمه‌ی Dim از کلمه ی Public نیز می توان استفاده کرد).

۳- **متغیرایستا** : برای تعریف این نوع متغیر به جای کلمه‌ی Dim از کلمه ی Static استفاده می کنیم وتفاوت آن با متغیر های دیگر این است که پس از خاتمه اجرای رویه ای که در آن تعریف شده مقدار آن از حافظه پاک نمی شود ودر اجرای بعدی نیز می توان از این مقدار استفاده کرد.

نکته : قوانین نام گذاری اشیاء و فرم ها مانند قوانین نامگذاری متغیرها می باشد.

✍ کنترل زمان سنج (**Timer**) : این کنترل در هنگام اجرا روی فرم دیده نمی شود، اگر بخواهیم عملیات یا دستوراتی را به طور خودکار و مکرر در طی زمان معینی اجرا کنیم از این کنترل استفاده می کنیم (دستورات را در رویداد timer آن می نویسیم وفقط همین یک رویداد را دارد).
دارای خاصیتی به نام Interval می باشد که زمان اجرا را مشخص می کند و بر حسب میلی ثانیه می باشد یعنی اگر Interval را برابر ۱۰۰۰ قرار دهیم هریک ثانیه یک بار دستورات داخل timer اجرا می شوند.

نکته : اگر بخواهیم به یک متغیر مقدار اولیه بدهیم این کار را در رویداد load فرم انجام می دهیم.

✍ خاصیت **Enabled** : از این خاصیت برای فعال یا غیر فعال کردن اشیاءو فرم ها استفاده می شود .

عملگرهای رشته‌ای:

از عملگر **جمع** (+) برای الحاق یا اتصال رشته ها استفاده می شود در واقع دو یا چند رشته را به یکدیگر اضافه می کند مثلا "ali" + "reza" برابر "alireza " می شود (به جای علامت + از علامت & نیز می توان استفاده کرد).

عملگر **like** : این عملگر دو رشته را باهم مقایسه کرده اگر هردو یکی باشندجواب آن True و اگر دو رشته یکسان نباشند جواب False خواهد بود.

مثال : "ali" Like "ali" جواب True

"ali" Like "reza" جواب False

"Ali" Like "ali" جواب False

به جای کارکتر ها از کارکتر های عمومی نیز استفاده می شود :

* به جای چند کاراکتر حرفی "ali" Like "a[d-n]i" جواب True

? به جای یک کاراکتر حرفی "aliReza" Like "al*" جواب True

به جای یک کاراکتر عددی "ali" Like "a?i" جواب True

[ ] بازه ای از کاراکتر ها "a257" Like "a#57" جواب True

علامت ! یعنی دراین بازه نباشد "ali" Like "a[!a-c]i" جواب True

تابع Str : با استفاده از این تابع مقادیر عددی به مقادیر رشته‌ای تبدیل می شوند (بر عکس تابع val)

نکته: اگر در پنجره کدنویسی(ماژول کد) برنامه در یک خط جا نشد انتهای خط علامت زیر خط _ را قرار داده و ادامه آن سطر دستور را در سطر بعدی می‌نویسیم.

خاصیت **Borderstyle** : از این خاصیت برای تنظیم نوع حاشیه دور اشیاء و فرم ها استفاده می شود.

واحدکار پنجم

توانایی استفاده از دستور شرطی IF و عملگرهای مقایسه‌ای و منطقی

ساختارهای تصمیم یکی از انواع دستورات vbمی باشد که به کاربر اجازه می دهد که نحوه‌ی اجرای دستورات یا برنامه را به میل خود و بر اساس شرط یا شرایط اجراکند یا اجرا نکند یکی از این نوع دستورات ، دستور **if** می باشد.

دستورIf : این دستور با بررسی شرط یا شرط های تعیین شده وبا توجه به درست (True) یا نادرست (False) نتیجه شرط‌ها روند اجرای برنامه یا دستورات را مشخص می کند.

شکل های کلی دستور if:

1 If شرط(شرط ها) Then دستور

2 If شرط(شرط ها) Then دستور۱ Else دستور۲

3 If شرط(شرط ها) Then

دستورات

End if

4 If شرط(شرط ها) Then

دستورات۱

Else

دستورات۲

End if

(در همه شکل های دستور if اگر نتیجه شرط True باشد دستور(ات) بعد از Then اجرا می‌شود و اگر False باشد دستور(ات) بعد از Else اجرا می‌شود.)

مثال: برنامه‌ای که یک عدد صحیح را توسط یک کادر متن دریافت و مشخص می‌کند زوج است یا فرد:

```
Private  Sub  Command1_Click( )
  Dim  x  as  Integer
  X= Val(Text1.text)
  If  x  mod  2 = 0  Then
      Print  "zoj  ast"
  Else
      Print  "fard  ast"
  End  if
End  Sub
```

برای قرار دادن تصویر روی فرم می توان از خاصیت picture فرم استفاده کرد و یا از کنترل های Picturebox و Image استفاده کرد.

کنترل کادر تصویر (**Picturebox**): این کنترل دارای خاصیتی به نام Picture می‌باشد که از آنجا مسیر و نام فایل گرافیکی (تصویری) را برای نمایش تعیین می‌کنیم . این کنترل دارای خاصیتی به نام Autosize می باشد که اگرمقدار آن True شود اندازه کنترل یا شیء با اندازه تصویر به طور خودکار

هماهنگ وبرابر خواهد شد واگر مقدار آن False شود اندازه کنترل (شیء) تغییری نمی کند وتصویر کامل نمایش داده نمی شود. (خاصیت Align در این کنترل محل نمایش تصویر را تعیین می کند.)

کنترل کادر تصویر علاوه بر رویدادهای مشترک با سایر کنترل ها دارای دو رویداد است: رویداد Change که زمانی رخ می دهد که تصویر نمایش داده شده توسط آن تغییر کند و رویداد Resize نیز زمانی که اندازه کنترل تغییر کند.

کنترل تصویر (Image) : این کنترل دارای خاصیتی به نام picture می باشد که از آنجا تصویر را انتخاب می کنیم، دارای خاصیتی به نام Stretch است که اگر True شود اندازه تصویر به اندازه کادر کنترل یا شی ء می شود و اگرFalse شود کادر کنترل به اندازه تصویر می شود .

برای آن که یک کلید دسترسی سریع (کلید میانبر) برای یک کنترل(CommandButton) ایجاد کنیم باید در خاصیت Caption قبل از کاراکتر (حرف) مورد نظر علامت & را قرار دهیم حال اگر هنگام اجرای برنامه کلید Alt را نگه داشته وحرفی را که زیر آن با این کار خط افتاده را بزنیم معادل کلیک کردن روی آن دکمه فرمان خواهد بود .

روش اضافه کردن فرم جدید به برنامه در واحد کار سوم توضیح داده شد.

متد show : این متد می‌تواند فرم را که اطلاعات آن در حافظه بار گذاری شده را نمایش دهد . Form2.Show

متدhide : این متد برعکس متد show می باشد واز نمایش فرم جلوگیری کرده وآن را مخفی می‌کند اما اطلاعات فرم را از حافظه خارج نمی کند.

Form2.Hide

خاصیت **Passwordchar** : این ویژگی مخصوص کنترل کادر متن (Textbox) می باشد وکاراکتر ی که در آن قرار می دهیم را به جای کاراکترهای تایپ شده در کادر متن نمایش می دهد (برای این که متن دیده نشود مناسب است مثلاًدر رمز ورود (پسورد)).

خاصیت **Maxlength** :این ویژگی مخصوص Textbox بوده و حداکثر تعداد کارکتر هایی را که در این کنترل می توان تایپ کرد را معین می‌کند . (مقدار پیش‌فرض آن صفر می باشد که تعداد کاراکتر های تقریبا نامحدودی(حداکثر تعداد ممکن) را قبول می کند).

خاصیت **Locked** : این ویژگی مخصوص Textbox می باشد واگر مقدار آن True شود کاربر توانایی ورود و ویرایش اطلاعات در آنرا نخواهد داشت یعنی کنترل Textbox قفل می شود.

نکته: دستور ""=Text1.text محتویات کادر متن Text1 را پاک می کند.

عملگرهای مقایسه‌ای:

برای مقایسه همه انواع داده ها می توان از این عملگرها استفاده کرد وفقط باید توجه داشت که دوعبارت یا داده که با هم مقایسه می شوند ازیک نوع داده باشند.

کوچکتر	<
بزرگتر	>
کوچکترمساوی	< =
بزرگترمساوی	> =
مساوی	=
نامساوی	< >

(تقدم همه یکسان است)

این عملگر ها در عبارات شرطی استفاده می گردند.

در مقایسه کاراکتر ها وعبارات رشته‌ای از کداسکی آنها برای مقایسه استفاده می شود. مثلا "ali" > "Ali" چون کد اسکی a (۹۷) از A(۶۵) بزرگتر است.

(در مقایسه رشته‌ها کاراکتر اول رشته اول با کاراکتر اول رشته دوم اگر یکی بودند دوم با دوم و . . . و به تعداد کاراکترهای رشته‌ها بستگی ندارد)

نکته: اگر درقسمت تعاریف عمومی(General)دستور Option compare text را بنویسیم vb بین حروف کوچک وحروف بزرگ در رشته‌ها تفاوتی قائل نمی‌شود(مثلا در اینصورت a با A تفاوتی ندارند).

در ویژوال بیسیک با اضافه کردن پیشوند vb به ابتدای اسامی برخی رنگ ها می‌توان به رنگ مورد نظر دسترسی داشت مثلا , VbBlue VbGreen , VbRed که به آنها ثابت های رنگی می گویند.

Form1.Backcolor = VbBlue

عملگرهای منطقی:

ازاین عملگر ها برای ترکیب شرط های مختلف استفاده می شود.

عملگر **And** ("و" منطقی):

اگر چند شرط با And ترکیب شوند در صورتی نتیجه کل عبارت Trueمی شود که همه شرط ها Trueباشند.

جدول صحت:

با مثال زیر این عملگر را بهتر خواهید فهمید:

(اگر توپ داشته باشیم (A) و (And) باران نیاید(B) آنگاه بازی می کنیم)

A	B	A and B
F	F	F
F	T	F
T	F	F
T	T	T

عملگر **OR** ("یا" منطقی):

اگر چند شرط با Or ترکیب شوند در صورتی نتیجه کل عبارت Trueمی شود که حداقل یکی از شرط ها Trueباشد.

جدول صحت:

با مثال زیر این عملگر را بهتر خواهید فهمید:

(اگر تاکسی باشد(A) یا(OR) اتوبوس باشد(B) آنگاه به مدرسه می‌رویم)

A	B	A and B
F	F	F
F	T	T
T	F	T
T	T	T

عملگر **Not** (نقیض):

عملگر Not ارزش یک شرط یا ترکیب شرط‌ها را معکوس می کند.

جدول صحت:

A	Not (A)
F	T
T	F

اولویت اجرای (تقدم) عملگرهای منطقی:

Not
And
Or

تقدم عملگرها بطور کلی:

۱ -عملگرهای ریاضی (محاسباتی)

۲ -عملگرهای اتصال رشته‌ها(+ و &)

۳ -عملگرهای مقایسه ای

۴ -عملگرهای منطقی

کنترل‌های آرایه‌ای: اگر نام (خاصیتName) چند کنترل را یکسان قراردهیم و برای خاصیت Index آنها مقادیر متفاوت از صفر شروع کنیم وبه آنها مقدار بدهیم در این صورت آرایه ای از کنترل‌ها ایجاد کرده ایم مثلا AB(0) , AB(1) , AB(2) و AB(3) که دراین مثال AB نام کنترل ها وعدد داخل پرانتز Index آنها می باشد.
(هنگامی که یک کنترل را Copy و Paste می کنیم یک کادر پیغام باز می شود اگر دکمه Yesرا بزنیم آرایه ای از کنترل ها ایجاد می شود و اگر No را بزنیم آرایه ایجاد نشده و برای شیء جدید نام جدیدی در نظر گرفته می‌شود، توضیح اینکه وقتی نام کنترل دوم را همنام با کنترل اول قرار می دهیم کادر پیغام فوق الذکر ظاهر می شود.)

تابع **Loadpicture**: با استفاده از این تابع می توان از طریق کدنویسی تصاویر یا عکس ها را در کنترل های Picturebox و Image قرار دارد (لود کرد):

شکل کلی: نام شیء . picture = Loadpicture (" نام و آدرس فایل گرافیکی ")
مثال: Image1 . picture = Loadpicture (" E:\aks\p1.jpg ")

نکته: اگر تابع را بصورت ("") Loadpicture استفاده کنیم تصویر نمایش داده در کنترل حذف می شود.
نکته : اگر بخواهیم رنگ زمینه کنترل Commandbutton را تغییر دهیم یا عکسی بر روی آن قرار دهیم ابتدا باید خاصیت Style آنرا برابر مقدار Graphical قرار دهیم و سپس خاصیت مورد نظر را تغییر دهیم.

خاصیت Default : مخصوص Commandbutton (دکمه فرمان) بوده اگر روی فرم چندین دکمه فرمان داشته باشیم فقط برای یکی از آنها مقدار این خاصیت را می توانیم True قرار دهیم که برای آن شیء زدن کلید Enter معادل با کلیک روی آن است ودستورات داخل رویداد Click آن اجرا می شود (حتماً نباید فوکوس روی آن شیء باشد).

خاصیت Cancel : مخصوص Commandbutton است اگر روی یک فرم چندین دکمه فرمان داشته باشیم فقط برای یکی از آنها مقدار این خاصیت را می توانیم برابر True قرار دهیم که برای آن شیء زدن کلید Esc روی کیبورد معادل کلیک روی آن شیء است ودستورات داخل آن اجرا می‌شود.

تعریف تابع : تابع مجموعه ای از دستورات است که مقادیر را دریافت و نتیجهعملیات را درجایی که از آن استفاده شده بازگشت می دهد.

تابع**Ltrim** : این تابع یک عبارت متنی (رشته ای) را دریافت و فاصله های خالی ابتدای آن (سمت چپ) را حذف می کند.
تابع **Rtrim** : این تابع یک عبارت متنی (رشته ای) رادریافت و فاصله های انتهای آن (سمت راست)را حذف می کند.
تابع **Trim** : این تابع فاصله های خالی هر دو طرف را همزمان حذف می کند.

مثال:
Ltrim(" valiasr") خروجی: "valiasr"
Rtrim("valiasr ") خروجی: "valiasr"
Ltrim(" valiasr ") خروجی: "valiasr"

کادر پیام (Msgbox) : در VB برای ایجاد کادر پیام از تابع Msgbox استفاده می شود :

شکل کلی:
1 **Msgbox** " متن پیام " , " عنوان " , دکمه‌ها و شکل آیکن و دکمه پیش‌فرض

2 Dim نام متغیر as Integer
نام متغیر = **Msgbox** (" متن پیام " , " عنوان " , دکمه‌ها و شکل آیکن و دکمه پیش‌فرض)

مثال:

```
Dim   A1  as   Integer
A1 = Msgbox ( "Do you want to Exit? " , vbYesNo + vbQuestion + vbDefaultButton2, "End")
```

با اجرای دستورات فوق کادر پیام مقابل ایجاد می‌شود:

نکته: بجای ثابت های رشته ای از ثابت‌های عددی هم می توان استفاده کرد مثلاً در مثال بالا بجای VbYesNo عدد ۴ را می توان قرار داد. بقیه این ثابت ها به صورت زیر می باشند:

نوع و تعداد دکمه ها	
ثابت رشته‌ای	ثابت عددی
VbOkonly	0
VbOkCancel	1
VbAbortRetryIgnore	2
VbYesNoCancel	3
VbYesNo	4
VbRetryCancel	5

نوع آیکن‌ها(شکل)		
ثابت رشته‌ای	ثابت عددی	شکل آیکن
Vbcritical	16	آیکن پیام خطای بحرانی
VbQuestion	32	آیکن پرسش
VbExclamation	48	آیکن پیام هشدار
VbInformation	64	آیکن پیام اطلاعات

دکمه پیش‌فرض		
ثابت رشته‌ای	ثابت عددی	توضیح
Vbcritical	0	اولین دکمه
VbQuestion	256	دومین دکمه
VbExclamation	512	سومین دکمه
VbInformation	768	چهارمین دکمه

دکمه پیش‌فرض دکمه‌ای است که فوکوس روی آن است

حال اگر بخواهیم با زدن یکی از دکمه ها کار خاصی یا دستورات خاصی انجام یا اجرا شوند می‌توانیم با استفاده از دستور if و ثابت‌های رشته‌ای یا عددی دکمه‌ها این کار را انجام دهیم مثلا اگر بخواهیم با زدن دکمه Yes در کادر پیام فرم بسته شود (در مثال قبل) دستور زیر را می‌نویسیم:

```
If    A1 = VbYes    Then    Unload  Me
```

بجای VbYes می‌توان 6 قرار داد

دکمه‌ای که کلیک می‌شود	
ثابت رشته‌ای	ثابت عددی
VbOk	1
VbCancel	2
VbAbort	3
VbRetry	4
VbIgnore	5
VbYes	6
VbNo	7

نکته: اگر نوع و تعداد دکمه‌ها در دستور Msgbox تعیین نشود بصورت پیش‌فرض دکمه Ok در کادر پیام نمایش می‌یابد.

نکته: اگر در دستور Msgbox عنوان تعیین نشود ، عنوان کادر پیام بصورت پیش‌فرض نام پروژه قرار می‌گیرد.

کادر ورود داده (Inputbox) : در VB علاوه بر کنترل کادر متن (Textbox) از تابع Inputbox که کادر ورود داده ایجاد می‌کند برای وارد کردن داده‌ها به برنامه استفاده می‌شود.

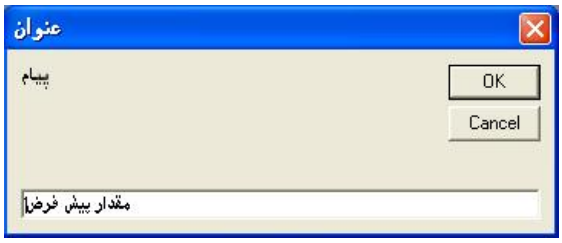
شکل کلی:

1 **Inputbox** " متن پیام ", " عنوان ", " مقدار پیش‌فرض ", x , y

۲ Dim نام متغیر as String

نام متغیر = **Inputbox** (" متن پیام ", " عنوان ", " مقدار پیش‌فرض ", x , y)

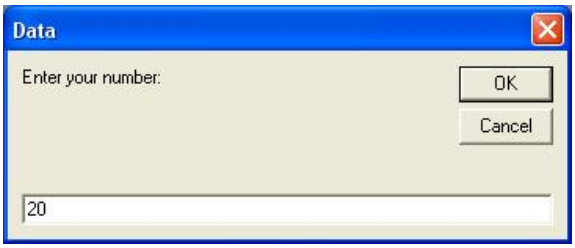
x فاصله کادر از سمت چپ صفحه دسکتاپ و y فاصله کادر از بالای صفحه دسکتاپ می‌باشند.



مثال:

```
Dim  x  as  String
x = Inputbox ( "Enter  your  number: " , "Data", "20")
```

با اجرای دستورات فوق کادر ورود داده مقابل ایجاد می‌شود:



خروجی تابع Inputbox رشته ای می باشد (یعنی همان داده‌ای که داخل کادر وارد می‌کنیم).

کنترل کادر علامت (Checkbox) :

از این کنترل برای ایجاد انتخاب های دو حالته استفاده می‌شود که در صورت انتخاب (تیک خوردن آن) و یا عدم انتخاب آن تصمیمات خاصی اتخاذ گردد و دستوراتی اجرا شوند یا نشوند.

خاصیت Alignment در این کنترل که باعث می شود اگر مقدار آن 0-Left Justify شود کادر علامت در سمت چپ عنوان کنترل و اگر مقدار آن 1- Right Justify شود کادر علامت در سمت راست عنوان آن نمایش می یابد.

خاصیت Style در این کنترل که باعث می شود اگر مقدار آن 0-Standard تنظیم شود شکل کنترل بصورت یک کادر علامت معمولی و اگر مقدار آن 1-Graphical Justify تنظیم شود به شکل یک دکمه فرمان دیده می شود.

خاصیت Value : اگر مقدار این خاصیت برابر 0-Unchecked باشد نشان دهنده اینست که کنترل انتخاب نشده است و اگر برابر 1-Checked باشد یعنی کنترل انتخاب شده و اگر برابر 2-Grayed باشد به این معنی است که کنترل دارای علامت چپ مارک (تیک) بوده و زمینه آن خاکستری است.

معرفی برخی خصوصیات

خاصیت **Appearance** : از این خاصیت برای تنظیم ظاهر اشیاء و فرم استفاده می‌شود، اگر مقدار آن 0-Flat شود مسطح و اگر 1-3D شود سه بعدی می شود.

خاصیت **Font** : برای تنظیمات قلم برخی اشیاء و فرم استفاده می‌شود.

خاصیت **MousePointer**: برای تعیین شکل اشاره گر موس برای برخی اشیاء یا فرم وقتی اشارگر روی آنها قرارمی گیرد ، استفاده می شود.

خاصیت **MouseIcon** : یک شکل آیکن خاص برای اشاره گر موس برای برخی اشیاء یا فرم تعیین می‌کند (ابتدا باید مقدار خاصیت Mouse Pointer را Custom قرار داد).

خاصیت **Visible** : برای نمایش یا عدم نمایش اشیاء بر روی فرم استفاده می شود.

خاصیت **TabIndex**: این خاصیت ترتیب دریافت فوکوس توسط کلید Tab برای اشیاء را مشخص می کند. (شی‌ئی که مقدار این خاصیت آن صفر باشد اولین فوکوس را دریافت می کند)

خاصیت **TabStop**: اگر برای شی‌ئی مقدار این خاصیت False شود آن شیء توسط کلید Tab فوکوس دریافت نمی‌کند .

خاصیت **MaxButton :** اگر مقدار آن True باشد دکمه Maximize فرم فعال و اگر False باشد غیر فعال می شود .

خاصیت **MinButton :** اگر مقدار آن True باشد دکمه Minimize فرم فعال و اگر False باشد غیر فعال می شود .

خاصیت **Icon** : این خاصیت یک شکل آیکن دلخواه را برای فرم تعیین می‌کند.

خاصیت **Tooltiptext** : از این خاصیت برای ایجاد کادر راهنمای متنی برای اشیاء استفاده می‌شود (زمانی که اشاره گر موس روی شیء مربوطه قرار گیرد عبارت موجود دراین خاصیت زیرا اشاره گر نمایش می یابد).

خاصیت **WordWrap** : این خاصیت مخصوص کنترل برچسب است و اگر محتویات متن آن (محتویات Caption) بیش از اندازه‌ی کنترل باشد و بخواهیم متن چندخطی شود مقدار این خاصیت را True می‌کنیم. (البته باید مقدار خاصیت Autosize را نیز True کنیم)

خاصیت **Moveable** : اگر مقدار این خاصیت True باشد فرم را می توان روی دسکتاپ جابجا کرد و اگر False باشد نمی‌توان آنرا جابجا کرد.

خاصیت **Windowstate :** مربوط به فرم می‌باشد حالت پنجره(فرم) را در هنگام اجرا مشخص می‌کند (اگر مقدار آن 0-Normal باشد فرم به شکل طبیعی و اگر 1-Minimized باشد بصورت دکمه و به حداقل رسیده در نوار وظیفه و اگر مقدار آن 2-Maximized باشد فرم با اندازه‌ی حداکثر نمایش داده می‌شود).

واحدکار ششم

توانایی استفاده از انواع حلقه ها و ساختار Select Case **و کنترل دکمه انتخاب**

یکی دیگر از دستورات ساختار های تصمیم select case می باشد

دستور select case : این دستور نیز مانند دستور if به شما این امکان را می دهد تا با توجه به شرایط مختلف دستورات مورد نظر را اجرا کنید در حالتی که نیاز به شرط ها و مقایسه های زیاد باشد دستور if مناسب نمی باشد و در این صورت از دستور select case استفاده می کنیم (این دستور زمانی که یک متغیر یا عبارت دارای مقادیر مختلفی باشد و بخواهیم بر اساس مقدار خاصی عملیات ویژه ای انجام شود استفاده می‌گردد).

شکل کلی دستور:

```
Select  case   متغیر یا عبارت مورد مقایسه
  Case     مقدار اول
           دستورات
  Case     مقدار دوم
           دستورات
  Case     مقدار سوم
           دستورات
      .
      .
      .
  Case  Else
           دستورات
End select
```

متغیر یا عبارات با مقدار اول مقایسه می شود اگر درست بود دستورات بعد از آن اجرا می شود در غیر این صورت با مقدار دوم مقایسه شده واگر درست بود دستورات آن اجرا می شود و به همین صورت الی آخر

(توضیح اینکه ممکن است متغیر یا عبارات در چندین مقدار شرط درست باشد در این صورت دستورات بعد از اولین شرط که درست(True) است فقط اجرا می شود)

استفاده از بخش Case Else اختیاری است.

نکته: در قسمت مقدار (یعنی بخش شرط)به سه شکل می توان آنرا نوشت:

الف) مقدار به صورت مستقیم نوشته شود مثلا Case 4

ب) بازه ای از مقادیر مثلا Case 1 to 5

ج) به صورت مقایسه ای مثلا case is<10

(در ضمن در حالت الف اگر بخواهیم بجای یک مقدار چند مقدار قرار دهیم آن مقادیر را با علامت کاما (,) استفاده می کنیم مثلا Case 4,5,10)

مثال: برنامه ای که معدل یک دانش آموز را دریافت می کند و اگر معدل بین ۱۵ تا ۲۰ بود وضعیت او را خوب اگر بین ۱۲ تا ۱۵ بود متوسط و اگر بین ۱۰ تا ۱۲ بود ضعیف و اگر کمتر از ۱۰ بود وضعیت او را بسیار ضعیف نمایش می دهد.

```
X= val(text1.text)
Select  Case  X
  Case   15  to  20
       Print   "خوب"
  Case   12  to  15
       Print   "متوسط"
  Case   10  to  12
       Print   "ضعیف"
  Case   is < 10
       Print   "بسیار ضعیف"
End  Select
```

✍ **کنترل OptionButton** : به این کنترل دکمه رادیویی و یا دکمه انتخاب نیز می گویند وبه شما اجازه می دهد تا امکان انتخاب یک گزینه از میان چند گزینه را داشته باشید و اصولا دو یا چند تا از این کنترل را روی فرم قرار می‌دهیم و فقط یکی از آنها را می توان انتخاب کرد
این کنترل دارای خاصیتی با نام **value** است که اگر مقدار آن True باشد یعنی آن کنترل (شی) انتخاب شده است و اگر False باشد یعنی انتخاب نشده است.
(توضیح اینکه از دکمه های انتخاب روی فرم فقط برای یکی از آنها مقدار خاصیت Value آن True می باشد و بقیه False هستند.)

✍ **کنترل قاب Frame :** از این کنترل برای گروهی کردن(دسته بندی) اشیا(کنترل‌ها) استفاده می شود(بخصوص برای گروهی کردن کنترل های OptionButton) یعنی اگر بخواهیم دو یا چند گروه مستقل از دکمه های انتخاب داشته باشیم و بصورت جداگانه عمل کنند از کنترل Frame برا ی اینکار استفاده می کنیم و هر گروه را داخل یک Frame قرار می دهیم.

✍ **ساختارهای تکرار(حلقه های تکرار):**

بعضی اوقات در برنامه ها لازم است برخی دستورات چندین بار تکرار شوند برای این کار از حلقه های تکرار استفاده می کنیم . که در بعضی از این نوع از دستورات تعداد دفعات تکرار مشخص است(مثل حلقه For) و در بعضی دیگر تعداد دفعات تکرار مشخص نیست وتکرار بر اساس شرایط معینی مشخص می شود.

شکل کلی : **For شمارنده حلقه = مقدار اولیه to مقدار نهایی Step گام حرکت**

دستورات

Next شمارنده حلقه

شمارنده حلقه یک متغیر می باشد

مثال:

```
For I=1 to 10 Step 1
    Print "Visual"
Next I
```

در این مثال دستور Print "Visual" ، ۱۰ بار تکرار می‌شود یعنی کلمه Visual ، ۱۰ بار چاپ می شود.

نکته: اگر گام حرکت ۱ باشد قسمت Step را می توان ننوشت.
نکته: در انتهای حلقه بعد از Next می توان شمارنده حلقه را ننوشت یعنی فقط Next قرار دارد.
نکته: اگر گام حرکت عددی مثبت باشد به حلقه،حلقه for افزایشی می گویند و اگر گام حرکت عددی منفی باشد به آن حلقه for کاهشی می گویند .
نکته: اگر گام حرکت مثبت باشد و اگر مقدار اولیه از مقدار نهایی بزرگتر باشد در این صورت حلقه اجرا نخواهد شد.
نکته: از رابطه زیر برای بدست آوردن تعداد دفعات اجرای دستورات داخل حلقه For می‌توان استفاده کرد:

مقدارنهایی - مقدار اولیه / گام حرکت

مثال هایی از حلقه For :

تکه برنامه ای که اعداد ۱ تا ۱۰ را چاپ می‌کند:

```
For X=1 to 10 Step 1
    Print X
Next
```

تکه برنامه ای که شکل مقابل را چاپ می‌کند: * * * * *

```
For X=1 to 5 Step 1
    Print "*" ;
Next
```

تکه برنامه ای که شکل مقابل را چاپ می‌کند:

* * * * *
* * * * *

```
For   I=1    to     2
   For   J=1    to     5
         Print  "*" ;
   Next
   Print
Next
```

تکه برنامه ای که شکل مقابل را چاپ می‌کند:

```
    *
    * *
    * * *
    * * * *
    * * * * *
```

```
For   i=1    to     5
   For   j=1    to     i
         Print  "*" ;
   Next
   Print
Next
```

تکه برنامه جدول ضرب ۱۰در ۱۰:

```
For   i=1    to     10
   For   j=1    to     10
         Print  i*j ;
   Next
   Print
Next
```

تکه برنامه ای که مجموع اعداد زوج طبیعی کمتر (یا مساوی) از n را چاپ می کند:

```
n = Val(Text1.text)
Sum = 0
For   X=2    to     n     Step 2
      Sum = Sum + X
Next
Print  Sum
```

✍**دستور Print :** از این دستور برای چاپ (نمایش) عبارات و مقادیر متغیرها بر روی فرم استفاده می شود. برای نمایش متن یا پیام آنها را عینا داخل دابل کوتیشن(") قرار می‌دهیم.

مثال:

```
Print  "Visual"
Print  X                مقدار داخل متغیر چاپ می شود
```

نکته: اگر از دستور Print به تنهایی استفاده شود یک خط خالی ایجاد می شود.

نکته: اگر از کاراکتر سمی‌کالن (;) استفاده می شود خروجی ها پشت سر هم و بدون فاصله نمایش داده می شود (نکته اینکه این علامت روی چاپ بعدی تاثیر می گذارد).

نکته: اگر از کاراکتر کاما(,) استفاده شود برای هر چاپ فضای ۱۴ تایی در نظر گرفته شده و در این فضا عبارت چاپ می شود و اگر خروجی از این مقدار بزرگتر باشد کل عبارت چاپ و چاپ بعدی یک ناحیه جلوتر نمایش می یابد.

مثال:

```
Print  "Ali"
Print  "Reza"        خروجی →
```

```
Ali
Reza
```

```
Print  "Ali";
Print  "Reza"        خروجی →
```

```
AliReza
```

```
Print  "Ali",
Print  "Reza"        خروجی →
```

```
Ali          Reza
```

```
Print  "Ali"; "Hadi",
Print  "Reza"        خروجی →
```

```
AliHadi          Reza
```

✍ تا کنون در مورد حلقه For که درآن تعداد دفعات تکرار معین می باشد صحبت گردید اما حلقه هایی وجود دارندکه تعداد دفعات تکرار دستورات در آنها مشخص نمی باشد و این حلقه ها دستورات را تا زمانی که شرط یا شرط های تعیین شده درست باشند (یا درست نباشند) تکرار می‌کنند.

شکل کلی:

1 While (شرط یا شرطها)

دستورات

Wend

2 Do While (شرط یا شرطها)

دستورات

Loop

در این دو نوع حلقه ابتدا شرط(ها) بررسی می‌شوند و اگر نتیجه True باشد (یعنی شرط برقرار باشد) آنگاه دستورات داخل حلقه اجرا می‌شوند و به همین صورت تکرار می شوند و اگر نتیجه False شود اجرای حلقه به پایان می‌رسد.

مثال: تکه برنامه ای که اعداد زوج کمترمساوی ۲۰ را چاپ می‌کند:

```
X=0
Do  While   (X <= 20)
   Print X
   X=X+2
Loop
```

3 Do Until (شرط یا شرطها)

دستورات

Loop

در این نوع حلقه ابتدا شرط(ها) بررسی می‌شوند و اگر نتیجه False باشد (یعنی شرط برقرار نباشد) آنگاه دستورات داخل حلقه اجرا می‌شوند و به همین صورت تکرار می شوند و اگر نتیجه True شود اجرای حلقه به پایان می‌رسد. (عکس حلقه Do While عمل می کند)

مثال: تکه برنامه ای که یک عدد طبیعی را دریافت و مجموع ارقام آنرا محاسبه و چاپ می‌کند:

```
X= Val(Text1.Text)
Sum = 0
Do  Until   (X = 0)
   D = X  mod  10
   X=X \ 10
   Sum = Sum + D
Loop
Print  Sum
```

4 Do

دستورات

Loop While (شرط یا شرطها)

5 Do

دستورات

Loop Until (شرط یا شرطها)

این دو نوع حلقه از نظر کارکرد مثل انواع ۳ و ۴ می باشند با این تفاوت که ابتدا یکبار دستورات داخل حلقه اجرا می‌شوند و سپس درستی یا نادرستی شرط ها در انتهای حلقه بررسی می‌شوند (بنابراین در این دو نوع، دستورات داخل حلقه حداقل یک بار اجرا می‌شوند).

اگر در نوار ابزار استاندارد روی دکمه Add Standaed EXE project کلیک کنیم یک پروژه جدید ایجاد می شود در واقع این پروژه جدید به پروژه قبلی اضافه شده و پروژه قبلی نیز هنوز باز است.

نکته: در صورتی که در یک پروژه بیش از یک فرم داشته باشیم و بخواهیم از ماژول کد یک فرم به متدها و خواص یک کنترل در یک فرم دیگر دسترسی پیدا کنیم به شکل زیر عمل می کنیم :

نام متد یا خاصیت . نام کنترل . نام فرم

این کد را در ماژول کد Form1 نوشته ایم

مثال : Form2 . Text1 . Forecolor = Vbred

گاهی نیاز است قبل از آنکه اجرای حلقه به طور عادی خاتمه پذیرد از حلقه خارج شویم در اینصورت از دستور Exit For برای خروج از حلقه For و از دستور Exit Do برای خروج از حلقه های Do while و Do Until استفاده می‌شود.

واحدکار هفتم

توانایی ایجاد و استفاده از انواع رویه‌ها در ویژوال بیسیک

در vb برنامه به صورت تکه تکه و قطعه برنامه ها می باشد و هر قطعه برنامه بطور مستقل می تواند اجرا و استفاده شود و به این صورت خطایابی و ایجاد تغییر در برنامه ها ساده تر می‌شود که به این روش برنامه نویسی ساخت یافته (Structural) می‌گویند و به این قطعه برنامه‌ها رویه (procedure)می گویند.

در ویژوال بیسیک چهار نوع رویه وجود دارد :

۱ - رویه های فرعی(Sub procedure).

۲ - رویه های تابعی(Function procedure)

۳ - رویه های رویدادی(Event procedure)

۴ - رویه های آماده ویژوال بیسیک (Visual Basic procedure)

رویه های رویدادی:

این نوع رویه ها بر اساس اشیا و رویدادها ایجاد می شوند و زمانی که رویداد مربوطه روی شیء مربوطه اتفاق بیفتد دستورات آن اجرا می شوند.

شکل کلی: Private sub Form یا نام شی_رویداد (آرگومان)

دستورات

End sub

آرگومان‌ها در واقع در متغیرهایی هستند که با توجه به رویداد اطلاعاتی را به داخل رویه ها وارد می‌کنند تا با استفاده از آنها مقاصد و اهدافی بر روی رویدادها انجام گیرد (مثلا آرگومانCancel در رویداد Unload از اتفاق افتادن آن جلو گیری میکند).

تا کنون رویداد های Click ،dblclick ، Load وChange معرفی شده اند، برخی دیگر از رویداد ها را معرفی می کنیم:

رویداد Unload : این رویداد مخصوص فرم است و زمانی که فرم بسته می شود یعنی از حافظه خارج می شود اتفاق می افتد.یک آرگومان به نام Cancel دارد که اگر مقدار آنرا ۱ (یا True) قرار دهیم از اتفاق افتادن این رویدا د جلو گیری می‌شود.

رویداد Activate : این رویداد مخصوص فرم است و زمانی که فرم فوکوس در یافت می‌کند(انتخاب می‌شود یا فعال می‌شود)اتفاق می‌افتد.

نکته: این رویداد بعد از رویداد Load اتفاق می‌افتد.

این رویداد هنگام انتقال فوکوس از پنجره یک برنامه به پنجره برنامه دیگر اجرا نخواهد شد.

رویداد Deactivate : این رویداد مخصوص فرم است وزمانی که فرم فوکوس از دست می‌دهد یعنی از انتخاب خارج میشود و غیر فعال میشود اتفاق می افتد.

رویداد GotFocus : این رویداد مخصوص فرم واشیا است وزمانی که فرم یا شی فوکوس را در اختیار می گیرد اتفاق می افتد.

نکته: هنگامی که یک فرم فوکوس بدست می‌آورد ابتدا رویداد Activate و سپس GotFocus اتفاق می‌افتد.

توضیح اینکه اگر هر دوی این رویدادها برای یک فرم استفاده شوند فقط Activate اجرا می‌شود و نیز اگر در فرمی کنترلی باشد که بتواند فوکوس بگیرد رویداد GotFocus برای آن فرم اجرا نمی‌شود.

رویداد LostFocus : این رویداد نیز مخصوص فرم و اشیا است و زمانی که فرم یا شی فوکوس را از دست می‌دهد اتفاق می افتد.

نکته: هنگامی که یک فرم فوکوس را از دست می‌دهد ابتدا رویداد LostFocus و سپس Deactivate اتفاق می‌افتد.

رویه های فرعی:

گاهی اوقات لازم است برنامه نویس دستورات مورد نظر را به صورت مستقل از قسمت های دیگر پروژه نوشته وهر جا نیاز شد از آنها استفاده کند در اینصورت از رویه های فرعی استفاده می‌شود.

این رویه ها را در بخش تعاریف عمومی (General) می نویسیم.

شکل کلی : Private sub نام رویه فرعی(نام آرگومان as نوع داده , . . .)

دستورات

End sub

برای فراخوانی و استفاده از رویه به شکل زیر استفاده می‌کنیم:

1 **Call نام رویه (آرگومان ها یا مقادیر)**

یا

2 **نام رویه آرگومان ها یا مقادیر**

نکته: برای تعریف رویه فرعی علاوه بر روش نوشتن مستقیم دستور می توان از منوی Tools وسپس گزینه Add procedure واز کادر باز شده با تنظیم آن رویه فرعی تعریف کرد.

مثال: رویه فرعی‌ای که زوج یا فرد بودن یک عدد را مشخص می‌کند:

```
Private  sub  zf (x  as  Integer)
  If  x mod 2 = 0  then
      Msgbox  "zoj"
  else
      Msgbox  "fard"
  End if
End sub
```

رویه های تابعی:

رویه های تابعی شبیه رویه های فرعی هستند با این تفاوت که از آنها در مواردی که نیاز باشد استفاده می‌شود ودر داخل آنها پس از دستورات، نتیجه (مقدار نهایی) به محل فراخوانی رویه بازگشت داده می‌شود (نتیجه را داخل نام رویه می ریزیم).

شکل کلی : Private Function نام تابع(نام آرگومان as نوع داده , . . .) as نوع مقدار بازگشتی تابع

دستورات

مقدار بازگشتی = نام تابع

End sub

مثال: رویه تابعی که فاکتوریل یک عدد را محاسبه می‌کند:

```
Private  Function  Fact (n  as  Integer) as Long
  Dim  x  as Integer , F  as long
  F=1
  For  x=1  to  n
      F = F * x
  Next
  Fact = F
End Function
```

روش فراخوانی و استفاده رویه تابعی همانند رویه فرعی می‌باشد و به شکل زیر نیز از آن می توان استفاده کرد:

نام متغیر= نام رویه تابعی(آرگومان ها یا مقادیر)

روش های ارسال مقادیر به رویه‌های فرعی و تابعی:

روش ارسال با مرجع: در این روش متغیر و آرگومان متناظر با آن به محل مشترکی اشاره می کنند بنابراین هر گونه تغییری در محتویات آرگومانها در رویه ، موجب تغییر در متغیرهای ارسالی می‌شود. در این نوع روش ارسال کلمه کلیدی ByRef در ابتدای نام آرگومانها در تعریف رویه ذکر می‌شود.

روش ارسال با مقدار: گاهی نیاز است بین متغیرهای ارسالی و آرگومانهای متناظر آنها ارتباطی بوجود نیاید و در صورت تغییر در مقدار آرگومانها در محتویات متغیرهای ارسالی تغییر حاصل نشود در این صورت از روش ارسال با مقدار استفاده می‌شود. در این نوع روش ارسال کلمه کلیدی ByVal در ابتدای نام آرگومانها در تعریف رویه ذکر می‌شود.

نکته: اگر در تعریف رویه نوع ارسال مشخص نشود نوع ارسال بطور پیش فرض از نوع ارسال با مرجع می‌باشد.

✍برای خروج از رویه فرعی قبل از اتمام دستورات آن از دستور Exit Sub و برای خروج از رویه تابعی قبل از اتمام دستورات آن از دستور Exit Function استفاده می شود.

✍اگر رویه با دستور Private تعریف شود رویه فرعی محلی می باشد یعنی فقط در ماژول کد فرمی که تعریف شده قابل شناسایی است واگر با دستور Public تعریف شود رویه فرعی عمومی می باشد و در سایر ماژول کدهای فرم های دیگر قابل شناسایی است.(برای فراخوانی رویه فقط کافیست نام فرمی که در آن تعریف شده را قبل از نام رویه بیاوریم مثلا Form1.Fact(5))

✍رویه ها را با استفاده از **ماژول کد**های مستقل نیز می توان تعریف کرد. ماژول کد(Code Module) یکی دیگر از اجزای پروژه ها در VB می باشد و فقط از دستورات تشکیل شده است (یعنی فقط صفحه کد نویسی دارد). در این صفحه (ماژول کد) اگر رویه فرعی یا تابعی را با دستور Private تعریف کنیم یک رویه محلی تعریف کرده ایم یعنی این رویه فقط در همین ماژول کد قابل شناسایی است واگر با دستور Public تعریف کنیم یک رویه ی عمومی تعریف کرده ایم ودر تمام قسمت های پروژه قابل شناسایی است.

نکته: از ماژول کدها برای تعریف متغیر های عمومی نیز استفاده می شود مثلا اگر دستور Public x as integer را در یک ماژول کد بنویسیم متغیر x در تمام ماژول کد فرم ها وسایر قسمت های پروژه شناسایی واستفاده می شود .

برای ایجاد ماژول کد ابتدا در پنجره ی پروژه کلیک راست کرده وسپس گزینه یAdd و زیرگزینه ی Module و دکمه ی Open را انتخاب می کنیم. (روش تعریف رویه ها و فراخوانی آنها در ماژول کد همانند تعریف آنها در ماژول کدهای فرم ها می باشد)

واحدکار هشتم

توانایی استفاده از انواع رویه‌های آماده در ویژوال بیسیک

در ویژوال بیسیک رویه های آماده متعددی وجود دارد که از قبل تعریف شده‌اند و در جاهایی که نیاز است فقط کافیست فراخوانی و استفاده شوند.

توابع رشته‌ای:

تابع **Asc** : این تابع یک مقدار رشته‌ای را دریافت و کد اسکی اولین کاراکتر آن را می‌دهد(مقداری بین 0 تا 255). Asc(رشته)

مثال	خروجی
Print Asc("Ali")	65
Print Asc("1")	49
Print Asc("computer")	99
Print Asc("")	خطا رخ می دهد

نکته: کد اسکی حروف بزرگ از ۶۵ (حرف A)شروع می‌شود و کد اسکی حروف کوچک از ۹۷ (حرف a)شروع می‌شود و ارقام از ۴۸ (کد کاراکتر صفر) شروع می‌شود.

تابع **Chr** : این تابع برعکس تابع Asc عمل می‌کند یعنی کد اسکی را دریافت و با توجه به آن کاراکتر معادل آنرا می‌دهد. Chr(کد اسکی)

مثال	خروجی
Print Chr(97)	a
Print Chr(50)	2
Print Chr(33)	!

نکته: خروجی عبارت Asc(Chr(65)) برابر 65 می باشد یعنی خود عبارت چون این دو تابع در واقع یکدیگر را خنثی می کنند.

تابع **Len** : این تابع یک رشته را دریافت و تعداد کاراکترهای آنرا نمایش می‌دهد(طول رشته). Len(رشته)

مثال	خروجی
Print Len("Visual")	6
Print Len("")	0
Print Len(نوع منطقی)	2

تابع **Left** : تعداد معینی کاراکتر را از سمت چپ یک عبارت رشته‌ای جدا می‌کند و برمی‌گرداند. Left(طول , رشته)

مثال	خروجی
Print Left("Visual",2)	Vi
Print Left("mohammad",4)	moha
Print Left("ali",7)	ali

تابع **Right** : تعداد معینی کاراکتر را از سمت راست یک عبارت رشته‌ای جدا می‌کند و برمی‌گرداند. Right(طول , رشته)

مثال	خروجی
Print Right("Visual",2)	al
Print Right("mohammad",4)	mmad

تابع **Mid** : این تابع تعداد معینی کاراکتر را از داخل یک عبارت رشته‌ای جدا می‌کند و برمی‌گرداند. Mid(رشته , شروع , طول)

مثال	خروجی
Print Mid("Visual",3,2)	su
Print Mid("mohammad",3)	hammad
Print Mid("ali",2,5)	li

نکته: اگر مقدار آرگومان شروع بزرگتر از تعداد کاراکترهای رشته باشد رشته ای با طول صفر بازگشت داده می‌شود.
اگر مقدار آرگومان طول که اختیاری است مشخص نشود تابع از نقطه شروع تا انتهای رشته را بازگشت می دهد.

تابع **Replace** : این تابع رشته ای را داخل رشته دیگر پیدا کرده و آنرا با رشته سوم جایگزین کرده و بصورت یک رشته جدید باز می‌گرداند.

Replace(string , find , replace , start , count , compare)

تابع رشته find را در رشته string پیدا کرده و رشته replace را جایگزین می کند. آرگومان start محل شروع جستجو در رشته string را معین می‌کند و اختیاری است و اگر استفاده نشود بصورت پیش فرض ۱ در نظر گرفته می شود. آرگومان count تعداد دفعات جستجو و جایگزینی را معین می‌کند و اختیاری است و اگر استفاده نشود بصورت پیش فرض ۱- در نظر گرفته می شود یعنی عمل جستجو و جایگزینی تا انتهای رشته انجام می شود. آرگومان compare اگر مقدار آن یک(یا vbTextCompare) باشد تابع در جستجو بین حروف کوچک و بزرگ تفاوت قایل نمی‌شود و اگر مقدار آن صفر(یا vbBinaryCompare) باشد تابع در جستجو بین حروف کوچک و بزرگ تفاوت قایل می‌شود (اگر این آرگومان استفاده نشود بصورت پیش فرض صفر می باشد)

مثال	خروجی
Print Replace("AlirezAhAmid","A","12")	12lirez12h12mid
Print Replace("BooOok","o","t",3,1,0)	tOok
Print Replace("abcab","a","")	bcb

نکته: اگر مقدار آرگومان replace یک رشته خالی("") باشد تمامی مواردی که در رشته پیدا می کند حذف می‌شوند.
اگر مقدار آرگومان start بزرگتر از یک باشد جواب از آن کاراکتر به بعد می باشد و کاراکترهای قبل آن حذف می شوند.

تابع **StrComp** : این تابع برای مقایسه دو رشته با یکدیگر استفاده می شود اگر دو رشته با هم مساوی باشند عدد صفر و اگر رشته اول از رشته دوم بزرگتر باشد عدد ۱ و اگر رشته اول از رشته دوم کوچکتر باشد عدد ۱- را بازگشت می دهد. آرگومان compare همانند تابع قبل است.

StrComp(string1 , string2 , compare)

مثال	خروجی
Print StrComp("Visual","visual",1)	0
Print StrComp("Ali","ali")	-1
Print StrComp("ali", "amir")	-1

تابع **InStr** : این تابع دو رشته را دریافت و موقعیت رشته دوم را در اولی پیدا کرده و اگر وجود داشت موقعیت آنرا بصورت یک عدد برگردانده و اگر وجود نداشت مقدار صفر را برمی گرداند.

InStr(start , string1 , string2 , compare)

آرگومان compare مثل قبل می باشد و آرگومان start نیز مثل قبل نقطه شروع جستجو را معین می کند.

مثال	خروجی
Print InStr(1,"cpu CPU CPU","CPU")	5
Print InStr(4,"alirezaAliAliali","ali",1)	8

نکته: اگر از دستور Option Compare Text در بخش تعاریف عمومی ماژول کد فرم استفاده شود در مقایسه ها توابع بین حروف کوچک و بزرگ تفاوت قایل نمی شوند (البته به شرطی که از آرگومان compare استفاده نکنند).

توابع رشته‌ای:

تابع **Date** : این تابع تاریخ سیستم را بر می گرداند و فاقد آرگومان است. با استفاده از این تابع به همراه دستور date می توان تاریخ سیستم را تنظیم کرد.

Print Date این دستور تاریخ سیستم را می دهد

Date = # تاریخ مورد نظر # این دستور تاریخ سیستم را تنظیم می کند(بجای # از " نیز می توان استفاده کرد)

تابع **Now** : این تابع تاریخ و ساعت سیستم را با هم بر می گرداند و فاقد آرگومان است.

Print Now

تابع **Day** : این تابع یک مقدار از نوع تاریخ را دریافت و شماره روز را که یک عدد بین ۱ تا ۳۱ است را برمی گرداند.

Print Day(Date) این دستور شماره روز جاری (طبق تاریخ سیستم) را می دهد

Print Day(#1/20/2012#) خروجی این دستور عدد20 است

تابع **Month** : این تابع یک مقدار از نوع تاریخ را دریافت و شماره ماه را که یک عدد بین ۱ تا ۱۲ است را برمی گرداند.

Print Month(Date) این دستور شماره ماه جاری (طبق تاریخ سیستم) را می دهد

Print Month(#2/20/2012#) خروجی این دستور عدد 2 است

تابع **Year** : این تابع یک مقدار از نوع تاریخ را دریافت و مقدار سال را برمی گرداند.

Print Year(Date) این دستور عدد سال جاری (طبق تاریخ سیستم) را می دهد

Print Year(#1/20/2012#) خروجی این دستور عدد 2012 است

تابع **MonthName** : این تابع شماره ماه های سال را دریافت و نام ماه متناظر با آن را بصورت یک عبارت رشته ای باز می گرداند.

MonthName (month , abbreviate)

آرگومان month شماره ماه است(۱ تا ۱۲) و آرگومان abbreviate که اختیاری و از نوع منطقی است اگر True باشد نام ماه بصورت خلاصه و اگر False باشد نام ماه را بطور کامل باز می گرداند(اگر استفاده نشود بطور پیش فرض False است) .

مثال	خروجی
Print MonthName (4,False)	April
Print MonthName (4,True)	Apr
Print MonthName (1)	January

تابع **WeelDay** : این تابع یک عبارت از نوع تاریخ را دریافت و یک عدد صحیح که بیانگر شماره روزهای هفته است را باز می گرداند(عددی بین ۱ تا ۷).

WeekDay (date , firstday)

آرگومان date تاریخ را وارد تابع می کند و آرگومان firstday که اختیاری است روز اول هفته را مشخص می کند (مقدار پیش فرض یکشنبه است).

آرگومان firstday		
روز هفته	**ثابت عددی**	**ثابت رشته‌ای**
یکشنبه	1	vbSunday
دوشنبه	2	vbMonday
سه شنبه	3	vbTuesday
چهارشنبه	4	vbWednesday
پنجشنبه	5	vbThursday
جمعه	6	vbFriday
شنبه	7	vbSaturday

خروجی	**مثال**
7	Print WeekDay (#3/10/2012#)
1	Print WeekDay (#3/10/2012#,vbSaturDay)

تابع **Time** : این تابع ساعت سیستم را بر می گرداند و فاقد آرگومان است. با استفاده از این تابع به همراه دستور Time می توان ساعت سیستم را تنظیم کرد.

Print Time این دستور ساعت سیستم را می دهد

Time = # ساعت مورد نظر # این دستور ساعت سیستم را تنظیم می کند(بجای # از " نیز می توان استفاده کرد)

تابع **Hour** : این تابع یک مقدار از نوع ساعت را دریافت و مقدار ساعت را که یک عدد بین ۰ تا ۲۳ است را برمی گرداند.

Print Hour(Time) این دستور ساعت جاری سیستم (طبق ساعت سیستم) را می دهد

Print Hour(#21:07:34#) خروجی این دستور عدد21 است

تابع **Minute** : این تابع یک مقدار از نوع ساعت را دریافت و مقدار دقیقه را که یک عدد بین ۰ تا ۵۹ است را برمی گرداند.

Print Minute(Time) این دستور دقیقه را با توجه به ساعت جاری سیستم می دهد

Print Minute(#21:07:34#) خروجی این دستور عدد7 است

تابع **Second** : این تابع یک مقدار از نوع ساعت را دریافت و مقدار ثانیه را که یک عدد بین ۰ تا ۵۹ است را برمی گرداند.

Print Second(Now) این دستور ثانیه را با توجه به ساعت جاری سیستم می دهد

Print Second(#21:07:34 AM#) خروجی این دستور عدد34 است

نکته: در سه تابع قبلی اگر مقادیر زمانی بزرگ تر از مقادیر مجاز باشند در هنگام اجرای برنامه پیام خطای Syntax error یا Type mismatch مشاهده می شود.

تابع **Timer** : با استفاده از این تابع می‌توان ثانیه های گذشته از نیمه شب (ساعت صفر) را بدست آورد.

✍ کنترل های کادر لیست (Listbox و Combobox)

این کنترل ها به کاربر اجازه می دهند تا از بین چند مقدار(گزینه) مختلف یکی را انتخاب کند. دو نوع می باشند: ۱- کادر لیست معمولی(Listbox)

۲- کادر لیست ترکیبی بازشو(Combobox) ، تفاوت آنها علاوه بر شکل ظاهری این است که Listbox فاقد کادر متن است اما Combobox از یک کادر لیست ساده و یک کادر متن تشکیل شده است.

برای ایجاد این لیست ها یعنی قرار دادن گزینه (عضو) برای این کنترل ها از دو روش می توان استفاده کرد:

الف) استفاده از خاصیت List یعنی در پنجره‌ی خواص در این خاصیت گزینه ها را وارد می‌کنیم و بعد از هر گزینه یک بار کلید ترکیبی Ctrl + Enter رامی زنیم تا به خط بعدی برویم.

ب) استفاده از از متد AddItem : با هر بار استفاده از این متد یک عضو به لیست اضافه می شود و شکل کلی آن به صورت زیر است:

نام کنترل لیست . AddItem("عضو")

یا

نام کنترل لیست . AddItem "عضو"

مثال: List1.AddItem "Blue"

نکته : برای اضافه کردن عضو به لیست توسط متد AddItem آنرا در رویداد load فرم می نویسیم.

نکته : کنترل Listbox دارای خاصیتی به نام selected است که با استفاده از این خاصیت وتنظیم مقدار آن به True گزینه ی مورد نظر را می توان به عنوان گزینه ی انتخاب شده پیش فرض تعیین کرد.

مثال: List1.Selected(2)=True

با اجرای این دستور سومین گزینه ی List1 یعنی گزینه ای که اندیس آن ۲ است به صورت پیش فرض انتخاب می شود.

خاصیت **ListIndex**: این خاصیت در کادر های لیست شماره‌ی عضو انتخاب شده را معین می‌کند (مقدار این خاصیت برای اولین گزینه صفر برای دومین گزینه ۱ و. . .)

خاصیت **ListCount** : از این خاصیت برای نمایش تعداد گزینه های کادر های لیست (Listbox,Combobox)استفاده می شود.

خاصیت **Sorted** : این خاصیت از نوع منطقی بوده و برای مرتب کردن لیست ها استفاده می‌شود یعنی اگر مقدار آن True شود لیست به صورت صعودی (از کوچک به بزرگ) مرتب می شود.

خاصیت **Style** : الف) در کنترل Listbox اگر مقدار آن 0-standard باشد کنترل به صورت یک کادر لیست ساده واگر مقدار آن 1-checkbox باشد قبل از هر عضو یک کادر علامت نیز ظاهر میشود.

ب) در کنترل Combobox این خاصیت انواع مختلفی از آن به کاربر ارائه می‌دهد ، اگر مقدار آن 0-Dropdown combo باشد کنترل به صورت یک کادر لیست کشویی(بازشو)به همراه یک کادر متن میشود واگر مقدار آن 1-Simple combo listباشد کنترل به صورت یک کادر لیست ساده به همراه یک کادر متن میشود(دکمه ی کشویی را ندارد وبا کلید های جهت دار روی گزینه ها جابجا می شویم) واگر مقدار آن 2-Dropdown list باشد کنترل به صورت یک کادر لیست باز شوی بدون کادر متن نمایش داده می شود.

خاصت **Locked**: این خاصیت مخصوص Combobox است (علاوه بر Textbox) و همانند کادر متن کنترل را قفل می کند یعنی نمی توان تغییر یا استفاده ای از آن کنترل داشت.

خاصیت **Text**: این خاصیت عنوان (متن)عضو انتخاب شده ی کنترل های لیست را در خود نگهداری می‌کند.

متد **Clear** : به وسیله‌ی این متد تمام اعضای موجود در کنترل های کادر لیست به صورت یکجا حذف می گردند.

مثال: List1.clear

متد **RemoveItem** : این متد هر یک از اعضای کنترل کادر لیست را بر اساس شماره‌ی اندیس آن به صورت تک تک حذف می کند.

مثال: Combo1.RemoveItem(1)

دومین عضو کادر لیست combo1 راحذف میکند

نکته: این دو کنترل دارای دو رویداد به نامهای click و change (مخصوص combobox) می‌باشند ، رویداد click زمانی که یکی از اعضای کادر لیست انتخاب شود اتفاق می افتد. رویداد change زمانی اتفاق می افتد که محتویات کادر متن کنترل combobox تغییر کند.

واحدکار نهم

نحوه استفاده از رویدادهای موس و صفحه کلید

رویدادهای ماوس:

رویدادهای ماوس در ویژوال بیسیک عبارتند از: Click ، Dblclick ، MouseDown ، MouseUp ، MouseMove ، DragDrop

رویداد **MouseDown** : این رویداد زمانی اجرا می شود که یکی از کلید های موس به پایین فشرده شود .

Private Sub نام شئ _ MouseDown (Button as Integer , Shift as Integer , X as single , X as single)

دستورات

End Sub

این رویداد دارای چهار آرگومان است:
آرگومان Button : کلیدی از موس که کاربر فشرده است را تعیین می کند که مقادیر این آرگومان به صورت زیر است:

ثابت رشته‌ای	ثابت عددی	توضیح
vbLeftButton	1	دکمه سمت چپ موس
vbRightButton	2	دکمه سمت راست موس
vbLeftButton	4	دکمه وسط موس

آرگومان Shift : این آرگومان مشخص میکند که در هنگام فشرده شدن دکمه های موس کدامیک از کلیدهای ترکیبی Alt ، Ctrl و Shift و یا ترکیبی از آنها فشرده شده است. مقادیر این آرگومان به صورت زیر است:

ثابت رشته‌ای	ثابت عددی	توضیح
vbShiftMask	1	کلید Shift
vbCtrlMask	2	کلید Ctrl
vbAltMask	4	کلید Alt
vbShiftMask + vbCtrlMask	3	کلیدهای Shift و Ctrl
vbShiftMask + vbAltMask	5	کلیدهای Shift و Alt
vbCtrlMask + vbAltMask	6	کلیدهای Ctrl و Alt
vbShiftMask + vbCtrlMask+ vbAltMask	7	کلیدهای Shift ، Alt و Ctrl

آرگومان های X و Y : موقعیت اشاره گر موس را هنگام فشردن کلیدهای آن مشخص می‌کند (X فاصله از چپ و Y فاصله از بالای فرم یا شیء مربوطه) .

رویداد **MouseUp** : این رویداد زمانی اجرا می شود که یکی از کلید های موس که به پایین فشرده شده، رها شود و بالا بیاید.

رویداد **MouseMove** : این رویداد زمانی اجرا می شود که اشاره‌گر موس روی فرم یا شیء (کنترل) مربوطه حرکت کند .

نکته: شکل کلی و آرگومان های رویدادهای MouseUp وMouseMove دقیقا مشابه رویداد MouseDown است.

مثال: برنامه ای که با فشردن دکمه چپ موس و کلید Shift روی فرم رنگ آنرا سبز و با فشردن دکمه راست موس و کلید Alt روی فرم عنوان آنرا به Visual تغییر می‌دهد :

```
Private Sub   Form _ MouseDown ( Button  as  Integer , Shift  as  Integer , X as single , X as single )
  If  Button = vbLeftButton  and  Shift = vbShiftMask   Then   Me.Backcolor = VbGreen
  If  Button = 2  and  Shift = 4   Then   Me.Caption = "Visual"
End Sub
```

رویداد **DragDrop** : این رویداد زمانی اجرا می شود که یک کنترل(شیء) در روی فرم با استفاده از اشاره‌گر موس جابجا شود (کشیدن و رها کردن) .

```
Private Sub   (یا فرم) نام شئ _ DragDrop ( Source  as  Control , X  as  single , X  as  single )
        دستورات
End Sub
```

آرگومان Source نام شیء که عملیات جابجایی روی آن انجام می شود را مشخص می کند و
آرگومانهای X و Y مختصات نقطه ای که عمل رها کردن در آنجا صورت گرفته را مشخص می‌کنند.

نکته: برای انجام عملیات انتقال یک کنترل روی فرم ابتدا باید مقدار خاصیت Dragmode را برای شیء مربوطه به مقدار 1-Automatic تغییر داد و دستور زیر را در رویداد DragDrop فرم نوشت :

```
Source . Move    X , Y
```

✍ کنترل خط (**Line**) : با استفاده از این کنترل می توان انواع افقی ، عمودی و مورب را رسم نمود.
خصوصیت های X_1 و Y_1 مختصات نقطه شروع و خصوصیت های X_2 و Y_2 مختصات نقطه انتهایی خط را تعیین می‌کنند.
خصوصیت Bordercolor رنگ خط و خصوصیت BorderWidth ضخامت خط را تعیین می‌کند.
خصوصیت Borderstyle سبک خط (خط چین ، خط نقطه و . . .) را مشخص می‌کند.

✍ کنترل شکل (**Shape**) : با استفاده از این کنترل می توان اشکال مختلف هندسی مثل دایره ، بیضی ، مربع و . . . را رسم نمود.
برخی خصوصیت های این کنترل بصورت زیر می باشد:
Shape : نوع شکل هندسی را تعیین می‌کند.
Backcolor : رنگ داخل شکل را مشخص می‌کند.
Backstyle : اگر روی مقدار 0-Transparent تنظیم شود شکل بصورت شفاف نمایش داده می شود و اگر روی 1-Opaque تنظیم شود شکل با رنگی که در خاصیت Backcolor تعیین شده نمایش می یابد.
Bordercolor : رنگ کادر دور شکل را مشخص می کند.
Borderstyle : سبک (نوع) کادر دور شکل را مشخص می کند.
Borderwidth : ضخامت کادر دور شکل را مشخص می کند.

نکته: واحد اندازه‌گیری در ترسیم اشکال و خط در روی فرم بطور پیش فرض Twip است و برای تغییر واحد اندازه‌گیری می توان از خصوصیت ScaleMode فرم استفاده کرد.

رویدادهای صفحه کلید :

✍ رویدادهای صفحه کلید در ویژوال بیسیک عبارتند از: KeyDown ، KeyUp و KeyPress

رویداد **KeyDown** : این رویداد زمانی اتفاق می افتد که یکی از کلیدهای کیبورد به پایین فشار داده شود.
شکل کلی:

```
Private Sub  نام شئ یا فرم _ KeyDown ( Keycode  as  Integer , Shift  as  Integer )

        دستورات

End Sub
```

آرگومان Keycode کد کلید فشرده شده را تعیین می‌کند و آرگومان Shift نوع کلید ترکیبی(Alt ، Ctrl و Shift) فشرده شده و یا ترکیبی از آنها با کلید اصلی را تعیین می‌کند.

رویداد **KeyUp** : این رویداد از نظر شکل کلی و آرگومانها دقیقا مشابه KeyDown است با این تفاوت که زمانی اتفاق می افتد که کلید فشرده شده رها شده و بالا بیاید.

مثال: برنامه ای که با فشردن کلید Enter با کلید ترکیبی Alt فرم بسته می شود.

```
Private Sub   Form _ KeyDown ( Keycode  as  Integer , Shift  as  Integer )
        If  Keycode = 13  and  Shift = vbAltMask   Then   Unload me
End Sub
```

(کد کلید Enter ، ۱۳ می باشد و کد کلید Esc ، ۲۷ می‌باشد)

مثال: توسط برنامه زیر می توان کد (Keycode) کلید فشرده شده را در عنوان فرم مشاهده کرد (برای بدست آوردن کد کلیدها مناسب است):

```
Private Sub   Form _ KeyDown ( Keycode  as  Integer , Shift  as  Integer )
        Me.Caption = Keycode
End Sub
```

رویداد **KeyPress** : این رویداد زمانی رخ می دهد که یکی از کلید های کیبورد زده شود (فشرده و رها شود).
شکل کلی:

```
Private Sub  نام شئ یا فرم _ KeyPress ( KeyAscii  as  Integer )

        دستورات

End Sub
```

آرگومان KeyAscii کد اسکی کلید فشرده شده را مشخص می‌کند (تفاوت آن با دو رویداد قبلی این است که بین حروف کوچک و بزرگ تفاوت قائل می شود یعنی مثلا کلید A در این رویداد دارای کد اسکی (KeyAscii) متفاوت است (مثلاً اگر A باشد ۶۵ و اگر a باشد ۹۷) در واقع کاراکتری عمل می کند).

نکته: در رویداد KeyPress کد کلید های رقمی روی قسمت تایپی و قسمت ماشین حسابی یکسان هستند در صورتی که در رویداد های Keydown و Keyup کد آن ها متفاوت است یعنی در این دو رویداد هر کلید روی کیبورد یک کد (keycode) مجزا دارد .

رویداد Keypress کلید های تابعی (F1 تا F12) ، جهتی ، ترکیبی و مانند این ها را پشتیبانی نمی کند اما کلید Ctrl را با برخی علائم پشتیبانی می‌کند و نیز از کلیدهای Enter ، Backspace و ESC پشتیبانی می‌کند.

مثال: در برنامه زیر با زدن کلید a عبارت Visual روی فرم چاپ می‌شود:

```
Private Sub   Form _ KeyPress ( KeyAscii  as  Integer )
        If   KeyAcsii = 97    Then    Print  "Visual"
End Sub
```

نکته : با زدن یک کلید به ترتیب ابتدا رویداد KeyDown و بعد رویداد KeyPress و سپس رویداد KeyUp اتفاق می افتد . (اولویت رویداد های صفحه کلید)

خاصیت **keypreview** : این خاصیت مخصوص فرم است و اگر مقدار آن را برابر True کنیم رویدادهای صفحه کلید مربوط به فرم نسبت به رویدادهای صفحه کلید سایر اشیاء اولویت پیدا می کنند و زودتر اجرا می شوند .
(توضیح این که در حالت عادی وقتی مقدار این خاصیت False است ، رویدادهای صفحه کلید کنترلی که فوکوس روی آن است اجرا می شود)

نکته : وقتی فوکوس روی کنترل دکمه فرمان (CommandButton) قرار می‌گیرد ، کلیدهای Enter و حرکت مکان نما نمی‌توانند رویدادهای صفحه کلید فرم و کنترل دکمه فرمان را اجرا نمایند.

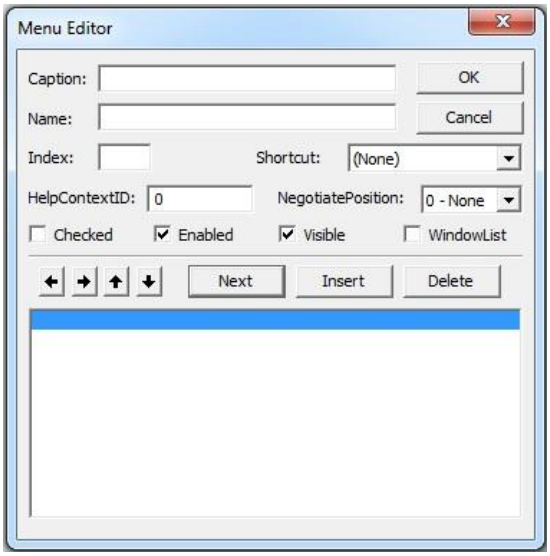
واحدکار دهم

نحوه ایجاد انواع منو در ویژوال بیسیک

نحوه طراحی منو در Vb :

برای ایجاد منوها در فرم ها از ابزار Menu Editor استفاده می‌گردد ، این ابزار به یکی از روش‌های زیر اجرا و باز می‌شود:

۱ – کلیک روی دکمه Menu Editor در نوار ابزار استاندارد
۲ – گزینه Menu Editor از منوی Tools
۳ – کلیک راست روی فرم و گزینه Menu Editor
۴ – کلید ترکیبی Ctrl + E



در کادر محاوره‌ای Menu Editor در قسمت Caption عنوان منو (گزینه) را وارد می‌کنیم (اگر همراه با عنوان از علامت & استفاده کنیم باعث می‌شود زیر کاراکتر بعد از آن خط قرار گرفته و با زدن کلید Alt همراه با آن حرف آن گزینه اجرا شود در واقع کلید ترکیبی ایجاد کرده‌ایم) . در قسمت Name یک نام برای آن گزینه قرار می‌دهیم (نکته اینکه اولا برای هر گزینه نام متفاوتی باید در نظر بگیریم ثانیا از قوانین نام‌گذاری متغیرها یا کنترلها باید تبعیت کنیم) با این کار یک منو (سرمنو) ایجاد کرده‌ایم، برای ایجاد گزینه های این منو کافیست مثل قبل عمل کرده و نیز با استفاده از دکمه → گزینه مورد نظر را به سمت داخل لیست ببریم با این کار یک زیرگزینه ایجاد کرده ایم .

از دکمه های ↑ و ↓ برای جابجایی گزینه ها استفاده می شود.

نکته: در کدنویسی هر گزینه با خاصیت Name آن شناخته می‌شود.

در Menu Editor از دکمه Next برای ایجاد و یا رفتن به گزینه بعدی منو و از دکمه Insert برای درج گزینه جدید(بین گزینه ها) و از دکمه Delete برای حذف گزینه(منو) استفاده می‌شود.

کادر Menu Editor دارای کادر علامت Visible است که اگر آنرا از انتخاب خارج کنیم (یعنی مقدار آن را False کنیم) گزینه مورد نظر مخفی خواهد شد و نیز خاصیت Checked اگر روی True تنظیم شود یعنی انتخاب شود باعث می‌شود که یک چک‌مارک (تیک) در کنار گزینه مورد نظر در منو ظاهر شود و نیز کادر علامت (خاصیت) Enabled باعث فعال یا غیرفعال شدن گزینه موردنظر می‌شود. از قسمت Shortcut نیز می‌توانیم کلید ترکیبی میانبر برای گزینه موردنظر در منو تعریف کنیم.

نحوه ایجاد رابط گرافیکی چند سندی (MDI):

تا کنون فرم‌هایی که ایجاد کردیم به طور مستقل از سایر فرم ها نمایش ، بسته ، منتقل. . . و استفاده می شدند و روی دیگر فرم ها تاثیری نمی گذاشتند که به این برنامه ها و فرم ها رابط گرافیکی تک سندی (فرم هایSDI) گفته می‌شود اما در برخی برنامه ها یک پنجره به عنوان پنجره‌ی اصلی و سایر فرم ها و پنجره ها در داخل این فرم باز می شوند وبه پنجره ی اصلی وابسته اند که به این نوع برنامه ها (فرم ها یا پنجره ها) رابط گرافیکی چند سندی یا MDI می‌گویند.
(به پنجره اصلی(فرم اصلی) پروژه فرم والد یا پدر (Parent) یا MDI و به فرم ها و پنجره های وابسته فرم فرزند (Child) گفته می‌شود.)

نکته: یک پروژه فقط یک فرم MDI می‌تواند داشته باشد.

نکته: استفاده از بعضی کنترل ها مثل دکمه فرمان ، کادر متن ، کادر تصویر و . . . در روی فرم های MDI امکان‌پذیر نیست اما کنترل هایی مثل Timer ، Data ، CommonDialog و . . . را می توان روی آنها قرار داد و استفاده کرد و نیز می توان روی آن منو ایجاد کرد.

برای **ایجاد** یک فرم از نوع MDI به یکی از روش های زیر عمل می‌کنیم:

۱ – گزینه Add MDI form از منوی Project
۲ – کلیک روی علامت مثلث دکمه Add Form در نوار ابزار استاندارد و انتخاب گزینه MDI form
۳ – کلیک راست در پنجره پروژه و گزینه Add و سپس گزینه MDI form
حال برای برقراری ارتباط و وابسته کردن یک فرم معمولی به فرم MDI خاصیت MDIchild فرم موردنظر را روی مقدار True تنظیم می‌کنیم.

کنترل های نوار پیمایش افقی و عمودی:

از این کنترل ها زمانی استفاده می‌شود که محتویات یک پنجره (فرم) از ابعاد آن بزرگ‌تر و بیشتر بوده و کاربر با استفاده از آنها می‌تواند محتویات آنرا مرور و نمایش دهد . در VB دو کنترل نوار پیمایش عمودی (VScrollBar) و نوار پیمایش افقی (HScrollBar) برای این منظور استفاده می‌شوند.

این دو کنترل دارای خواص و رویدادهای زیر هستند:

خصوصیت **LargeChange** : این خصوصیت مقدار تغییرات را هنگامی که روی مکانی از نوار پیمایش بجز دکمه ها (فضای خالی) کلیک می‌شود را معین می‌کند.

خصوصیت **SmallChange** : این خصوصیت مقدار تغییرات را هنگامی که روی دکمه‌های مثلثی نوار پیمایش کلیک می‌شود را تعیین می‌کند.

خصوصیت **Min** : این خصوصیت مقدار حداقل را وقتی دکمه متحرک ابتدای نوار است را مشخص می‌کند (معمولا صفر می‌باشد).

خصوصیت **Max** : این خصوصیت مقدار حداکثر را وقتی دکمه متحرک به انتهای نوار می‌رسد را مشخص می‌کند (معمولا مقدار آنرا به اندازه‌ی مقدار جابجایی مورد نیاز تعیین می‌کنند مثلا به اندازه مقداری که محتویات از فرم بزرگ تر است).

خصوصیت **Value** : این خصوصیت مقدار جاری را در نوار پیمایش معین می‌کند. (توضیح اینکه حداقل مقدار آن به اندازه Min و حداکثر مقدار آن به اندازه Max است)

رویداد **Change** : این رویداد زمانی رخ می‌دهد که مقدار خصوصیت Value تغییر کند یعنی زمانی که روی دکمه‌های مثلثی یا در مکان خالی روی نوار پیمایش کلیک کنیم.

رویداد **Scroll** : این رویداد زمانی رخ می‌دهد که کاربر دکمه متحرک را در نوار پیمایش بوسیله موس جابجا کند.

مثال : برای جابجایی و پیمایش کنترل تصویر Image1 که اندازه آن از فرم بزرگتر است کدهای زیر را می‌نویسیم:

```
Private Sub Form_Load()
    HScroll1.Min = 0
    HScroll1.Max = Image1.Width - Me.Width
    VScroll1.Min = 0
    VScroll1.Max = Image1.Height - Me.Height
End Sub

Private Sub HScroll1_Change()
    Image1.Left = -HScroll1.Value
End Sub

Private Sub HScroll1_Scroll()
   Image1.Left = -HScroll1.Value
End Sub

Private Sub VScroll1_Change()
   Image1.Top = -VScroll1.Value
End Sub

Private Sub VScroll1_Scroll()
  Image1.Top = -VScroll1.Value
End Sub
```

کنترل کادر محاوره (CommonDialog):

از این کنترل برای ایجاد کادرهای محاوره‌ای برای باز کردن و ذخیره‌سازی فایل ها ، انجام عملیات چاپ ، انتخاب رنگ ها و فونت ها و نمایش راهنما استفاده می‌شود.

این کنترل بصورت پیش فرض در ToolBox دیده نمی‌شود و برای افزودن آن به جعبه ابزار بصورت زیر عمل می‌کنیم:

از منوی Project گزینه Components را انتخاب تا کادر مربوط به آن باز شود (یا روی ToolBox کلیک راست و از منوی باز شده گزینه Components را زده و یا از کلید میانبر Ctrl+T استفاده کنید) سپس در پنجره باز شده در زبانه Controls از لیست گزینه Microsoft Common Dialog Conrol 6.0 را تیک می‌زنیم و ok را زده تا کنترل به جعبه ابزار اضافه گردد.

حال می توانیم کنترل CommonDialog را به فرم اضافه کرده و از آن استفاده کنیم ، این کنترل هنگام اجرای برنامه روی فرم دیده نمی‌شود، این کنترل دارای متدهای زیر برای نمایش کادرهای محاوره ای است:

نام کادر محاوره	نام متد
کادر محاوره باز کردن فایل‌ها	ShowOpen
کادر محاوره ذخیره‌سازی فایل‌ها	ShowSave
کادر محاوره رنگ ها	ShowColor
کادر محاوره فونت	ShowFont
کادر محاوره چاپگر	ShowPrinter
کادر محاوره راهنما	ShowHelp

مثال ۱: برنامه ای که با استفاده از کادر محاوره‌ای رنگ ها رنگ زمینه فرم را تغییر می‌دهد:

```
Private Sub Command1_Click()
   CommonDialog1.ShowColor
   Me.BackColor = CommonDialog1.Color
End Sub
```

مثال ۲: برنامه ای که با استفاده از کادر محاوره‌ای Open تصویر دلخواهی را در کنترل تصویر Image1 نمایش می‌دهد:

```
Private Sub Command1_Click()
  CommonDialog1.ShowOpen
  Image1.Picture = LoadPicture(CommonDialog1.FileName)
End Sub
```

برخی خصوصیات کنترل CommonDialog :

خصوصیت **FileName** : این خصوصیت نام و مسیر فایلی که انتخاب شده است را نگهداری می‌کند.

خصوصیت **FileTitle** : توسط این خصوصیت می‌توان به نام فایل انتخاب شده دسترسی پیدا کرد. (فقط نام فایل بدون مسیر آن)

خصوصیت **InitDir** : توسط این خصوصیت می‌توان مسیر پیش فرض(جاری) کادرهای Open و Save as را تغییر داد :

نام کنترل CommonDialog . InitDir = مسیر

بطور مثال اگر بخواهیم در مثال ۲ مسیر پیش فرض پوشه aks از درایو D باشد کد زیر را در رویداد Load فرم وارد می‌کنیم:

```
CommonDialog1 . InitDir  = "D:\aks"
```

خصوصیت **Filter** : توسط این خصوصیت می‌توان نوع فایل های مورد نمایش در کادر محاوره را تعیین کرد :

نام کنترل CommonDialog . Filter = "توضیح نوع فایل | پسوند فایل.* | . . . "

بطور مثال اگر بخواهیم در مثال ۲ فایل های (تصاویر) با فرمت bmp و jpg در کادر محاوره نمایش و باز شوند کد زیر را در رویداد Load فرم وارد می‌کنیم:

```
CommonDialog1 . Filter  = "Bitmap files(*.bmp) | *.bmp | JPEG(*.jpg) | *.jpg"
```

خصوصیت **FilterIndex** : اگر بیش از یک نوع فایل برای خصوصیت Filter تعیین شود بطور پیش فرض اولین نوع فایل ، فایل پیش فرض کادر محاوره قرار می‌گیرد توسط این خصوصیت می‌توان گزینه پیش فرض را تعیین کرد که اولین گزینه(نوع فایل) مقدار ۱ بعدی مقدار ۲ و الی آخر.

بطور مثال اگر بخواهیم در مثال ۲ گزینه دوم یعنی نوع فایل JPEG(*.jpg) در کادر محاوره گزینه پیش فرض باشد کد زیر را در رویداد Load فرم وارد می‌کنیم:

```
CommonDialog1.FilterIndex = 2
```

نکته: خصوصیات فوق را می توان از طریق کادر محاوره Property Pages نیز تنظیم کرد ، برای دسترسی به آن در پنجره خصوصیات ، کنترل CommonDialog مربوطه را انتخاب و از قسمت Custom این کادر باز می‌شود.