

به نام پروردگار

برنامه‌نویسی گام به گام با

ویژوال بیسیک

سوم راهنمایی

سید حامد یعقوبی شهریار

محمد سیاحت‌گر

مجموعه کتاب‌های تکمیلی سمپاد

کتاب سوم



نشر سمپاد



نشر سمپاد

برنامه‌نویسی گام به گام با ویژوال بیسیک

پایه سوم راهنمایی

مجموعه کتاب‌های تکمیلی سمپاد، کتاب سوم

نویسنده‌گان: سید حامد یعقوبی شهیر - محمد سیاحت‌گر

طرح جلد، صفحه آرایی و تصویرسازی: ضیا جلالی

حروف چینی: نشر سمپاد

شمارگان: ۱۰/۰۰۰ جلد

چاپ اول: ۱۳۸۶

چاپ و صحافی: طیف نگار

قیمت: ۱۰۰۰ تومان

کلیه حقوق برای نشر سمپاد محفوظ است.

شابک

ISBN

۶ مقدمه
۷	گام اول - مقدمه‌ای بر ویژوال بیسیک
۸ آشنایی
۹ ویژگی‌ها
۱۰ تمرین
۱۱	گام دوم - آشنایی با محیط ویژوال بیسیک و اولین برنامه
۱۲ آشنایی
۱۳ تغییر عنوان فرم
۱۴ خصوصیت (Property)
۱۵ آشنایی با جعبه‌ی ابزار
۱۶	اولین برنامه در ویژوال بیسیک
۱۷	گام سوم - یک ماشین حساب ساده
۱۸ آشنایی
۱۹	برنامه‌ی جمع دو عدد
۲۰	چگونه کار خود را ذخیره کنیم؟
۲۱ طراحی پروژه
۲۲	گام چهارم - دوره‌ای بر دستورات کنترلی زبان بیسیک
۲۳ آشنایی
۲۴	ساختارهای شرطی
۲۵ If ... Then
۲۶ If ... Then ... Else
۲۷	ساختارهای حلقه
۲۸ For ... Next
۲۹ While ... Wend
۳۰ تمرین
۳۱	گام پنجم - دوره‌ای بر گرافیک
۳۲ آشنایی
۳۳	جعبه‌ی تصویر (Picture Box) چیست؟
۳۴	قرار دادن تصویر در جعبه‌ی تصویر
۳۵ خصوصیت AutoSize
۳۶ خصوصیت ScaleMode
۳۷ خصوصیات Height و Width
۳۸ خصوصیت AutoRedraw
۳۹	دستور Pset

۵۲	دستور Line
۵۳	دستور Circle
۵۴	دستور Point
۵۴	درباره‌ی رنگ‌ها بیشتر بدانیم
۵۶	گام ششم - آشنایی با انواع متغیر در ویژوال بیسیک
۵۷	آشنایی
۵۷	انواع داده‌ای
۵۸	حوزه‌ی متغیرها
۶۳	دوره‌ی حیات (طول عمر) متغیرها
۶۶	گام هفتم - گرافیک ۱ (چرخش تصاویر).
۶۷	آشنایی
۶۷	پروژه ۱
۶۸	پروژه ۲
۷۰	گام هشتم - کار کردن با ماوس
۷۱	آشنایی
۷۱	خصوصیت MousePointer برای تغییر ظاهر اشاره‌گر ماوس
۷۲	انواع رخدادها
۷۳	نکاتی درباره رخدادهای MouseUp ، MouseMove و MouseDown
۷۶	تمرین
۷۷	گام نهم - گرافیک ۲ (سیاه و سفید کردن تصاویر)
۷۸	آشنایی
۸۲	گام دهم - کار کردن با صفحه کلید
۸۳	آشنایی
۸۳	رخدادهای صفحه کلید
۸۳	KeyPress
۸۴	KeyPress و KeyDown
۸۴	تفاوت بین دو رخداد
۸۵	KeyUp
۸۵	رخداد
۸۶	تمرین
۸۷	گام یازدهم - گرافیک ۳ (بزرگنمایی تصاویر)
۹۰	گام دوازدهم - توابع و زیربرنامه‌های تعریف شده توسط کاربر
۹۱	آشنایی
۹۲	تفاوت‌های بین برنامه و زیربرنامه
۹۵	نحوه‌ی تعریف زیربرنامه

۹۶	نحوه‌ی فراخوانی و یا استفاده از زیربرنامه
۹۸	نحوه‌ی تعریف تابع
۱۰۰	نحوه‌ی فراخوانی و یا استفاده از تابع
۱۰۱	تمرین
۱۰۲	گام سیزدهم - گرافیک ۴ (فیلترهای رنگی)
۱۰۳	آشنایی
۱۰۴	فیلتر اول - شطربنچی کردن تصویر
۱۰۵	فیلتر دوم - تنظیم روشنایی تصویر
۱۰۶	فیلتر سوم - ترکیب دو تصویر
۱۰۷	گام چهاردهم - کار کردن با فایل‌ها
۱۰۸	آشنایی
۱۰۹	فایل‌های متنی
۱۱۰	نحوه‌ی کار کردن با فایل‌های متنی در ویژوال بیسیک
۱۱۱	باز کردن فایل با دستور Open
۱۱۰	بستن فایل با دستور Close
۱۱۱	نوشتن در فایل متنی با کمک دستور #Print
۱۱۲	خواندن از فایل متنی با دستورهای #Input
۱۱۴	خواندن از فایل به کمک دستور Line Input#
۱۱۵	آشنایی بیشتر با فایل‌ها
۱۱۵	نوشتن در فایل متنی با کمک دستور #Write
۱۱۶	شماره‌ی فایل و دستور FreeFile
۱۱۶	تابع LOF
۱۱۷	دستور Kill
۱۱۷	کاراکترهای مهم در فایل‌های متنی
۱۱۷	تمرین
۱۱۸	پیوست یک - مقدمه‌ای بر شبکه‌های رایانه‌ای
۱۱۹	آشنایی
۱۱۹	مفاهیم پایه‌ی شبکه‌های رایانه‌ای
۱۲۰	مزایای شبکه
۱۲۱	انواع شبکه
۱۲۲	آشنایی با بعضی اصطلاحات دنیای شبکه
۱۲۳	چگونه نام یک کامپیوتر را در شبکه پیدا کنیم؟
۱۲۴	امکانات ویژوال بیسیک برای برنامه‌نویسی شبکه
۱۲۷	پیوست دو - جدول اسکی (ASCII)

مقدمه

« فقط یک چیز می‌دانم، و آن این‌که چیزی نمی‌دانم »

سفره

امروزه رایانه در اکثر فعالیت‌های بشر نقش اساسی پیدا کرده است و به عنوان یک یار و همراه محسوب می‌شود. این موضوع در تمامی رشته‌های علمی و عملی به روشنی قابل مشاهده است. البته همه این فعالیت‌ها صرفا شامل برنامه‌نویسی و برنامه‌سازی نیستند، اما نظر بسیاری افراد متخصص بر این است که آموزش برنامه‌نویسی و برنامه‌سازی دیدی منطقی و تحلیلی همراه با خلاقیت را برای دانش‌آموز به وجود می‌آورد. بی‌شک به همین دلیل است که در بسیاری از مدارس کشور از جمله مدارس تحت نظر سازمان ملی پژوهش استعدادهای درخشان، درس رایانه جایگاه مناسبی پیدا کرده است.

برنامه‌نویس دید گسترده‌ای به دنیای اطراف خود دارد. او می‌آموزد که چگونه می‌تواند در حل مسائل پیچیده و بزرگ از این اختراع بی‌نظیر بشر کمک بگیرد. او می‌آموزد که چگونه می‌تواند مسئله پیچیده خود را تبدیل به مسائلی ساده‌تر نموده تا بتواند به راحتی آن‌ها را از پیش رو بردارد.

در این کتاب سعی شده است تا از مطالب ساده، جذاب و در عین حال مفید استفاده شود. همچنین نحوه قرارگیری مطالب به‌گونه‌ای باشد که نوآموز به راحتی آن‌ها را دربال کرده و بتواند گام به گام با کتاب پیش روی و لی بی‌شک راهنمایی‌های معلم تاثیر بهسازی در پیشرفت او خواهد داشت.

کتاب حاضر نتیجه سال‌ها تجربه یادگرفتن و یاد دادن است. تجربه‌ای گران‌بها که از نسلی به نسل دیگر منتقل می‌شود. بی‌شک کتاب حاضر بیش‌تر از هرچیز، مرهون زحمات و تجربیات اساتید گرامی ماست. کسانی که لحظات گران‌قدر عمر خویش را برای تعلیم و تربیت ما صرف نمودند، تقدیم سپاس به تمامی این بزرگ مردان.

لازم به ذکر است که طراحی و تدوین محتوای درسی حاضر در طی جلسات متعدد گروهی و آزمون‌های میدانی انجام و در قالب محتوای الکترونیکی در طول سه سال تحصیلی متولی در مرکز علامه حلی تهران با موفقیت اجرا و پیاده‌سازی شده است. بر خود لازم می‌دانم که از آقایان محمد رضا جهانگیر، محمد سیاحت‌گر و احسان شهشهانی که طراحی و تدوین محتوای الکترونیکی بخش‌های بسیاری از کتاب را بر عهده داشتند قدردانی نمایم. همچنین از آقای محمد رضا جهانگیر که زحمت ویراستاری کتاب را پذیرفتد، نهایت سپاس‌گزاری را دارم.

امیدوارم که در تدوین این کتاب اشتباهاتم کمینه باشد، زیرا همان‌گونه که بیهقی می‌گوید: «احتیاط باید کرد نویسنده‌گان را در هرچه نویسنده، که از گفتار باز توان ایستاد و از نوشتن باز نتوان.» بدیهی است به رغم کوشش و دقت صورت گرفته، کتاب حاضر خالی از نقص نمی‌باشد. لذا از معلمان و دانش‌آموزان تقاضا دارم نظرات خود را با نشر سمپاد در میان گذاشته و ما را در ارائه کاری بهتر برای ویرایش‌های بعدی یاری دهن. پیشاپیش مراتب امتحان خود را تقدیم می‌دارم.

دانش‌آموخته‌ی سمپاد

سید حامد یعقوبی شهیر

تابستان ۱۳۸۶

گام اول

مقدمه‌ای بر زبان ویژوال بیسیک

۱.۱ آشنایی

ویژوال بیسیک یک زبان برنامه‌نویسی تحت سیستم عامل ویندوز است. این زبان برگرفته از زبان بیسیک می‌باشد که ممکن است با آن آشنا باشید. قبل از شروع کار در محیط برنامه‌نویسی ویژوال بیسیک، بهتر است ببینیم که چرا این زبان را برای یادگیری انتخاب کردایم.

زبان بیسیک در اواسط دهه‌ی ۶۰ میلادی، با هدف ارتقاء و گسترش دانش برنامه‌نویسی در بین دانشجویان، دانشجویان و یا هر فردی که می‌خواست پا به عرصه‌ی برنامه‌نویسی بگذارد، تدوین و طراحی شد. این زبان در طی سال‌های متتمادی با تغییرات و بهبودهای بسیاری همراه بوده است و همواره طراحان سعی در بالا بردن قابلیت‌های آن در حد یک زبان سطح بالا داشته‌اند؛ به طوری که در ویرایش‌های آخر آن، قابلیت‌های بسیاری برای برنامه‌نویسی وجود دارد. البته نباید فراموش شود که در تمامی ویرایش‌های زبان برنامه‌نویسی بیسیک، سادگی و سهولت کار، یک اصل اساسی در طراحی و ایجاد این زبان بوده است.

با ظهور سیستم‌عامل ویندوز، شرکت‌ها و موسسات تولید زبان‌های برنامه‌نویسی، بر آن شدند تا به سمت تولید زبان‌های برنامه‌نویسی تحت ویندوز حرکت کنند. فقدان زبانی مشابه با بیسیک که علاوه بر سادگی، دارای امکانات قدرتمند و همچنین کافی برای برنامه‌نویسی باشد، منجر به طراحی و تولید ویرایش اول زبان ویژوال بیسیک، در سال ۱۹۹۱ توسط شرکت مایکروسافت شد. به تدریج تولید ویرایش‌های بعدی ادامه یافت، تا این که ویرایش ششم این زبان محبوب و قدرتمند برنامه‌نویسی، در سال ۱۹۹۸ میلادی عرضه شد. ما نیز امسال به بررسی قابلیت‌های این ویرایش از زبان ویژوال بیسیک می‌پردازیم، امیدواریم در پایان این سال تحصیلی، همگی شما توانایی نوشتن برنامه‌های مورد علاقه‌ی خود را با این زبان داشته باشید.



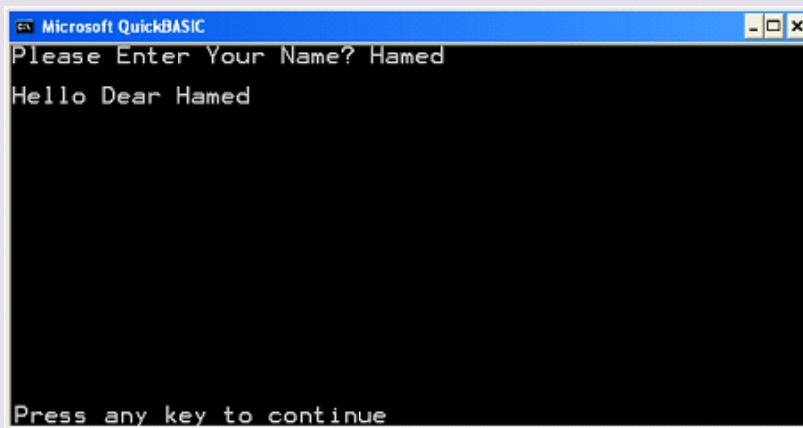
۱.۲. ویژگی‌ها

اکنون قصد داریم بعضی از ویژگی‌های زبان ویژوال بیسیک را به طور اجمالی مورد بررسی قرار دهیم:

- **رابط گرافیکی کاربر:** یکی از مزایای اصلی زبان ویژوال بیسیک، رابط گرافیکی کاربر است که امکان تولید سریع و آسان برنامه‌های تحت ویندوز (که اغلب دارای رابط گرافیکی کاربر پسند هستند) را فراهم می‌آورد. در جلسات آینده، بیشتر با محیط برنامه‌نویسی ویژوال بیسیک و نحوه‌ی استفاده از آن آشنا خواهیم شد.

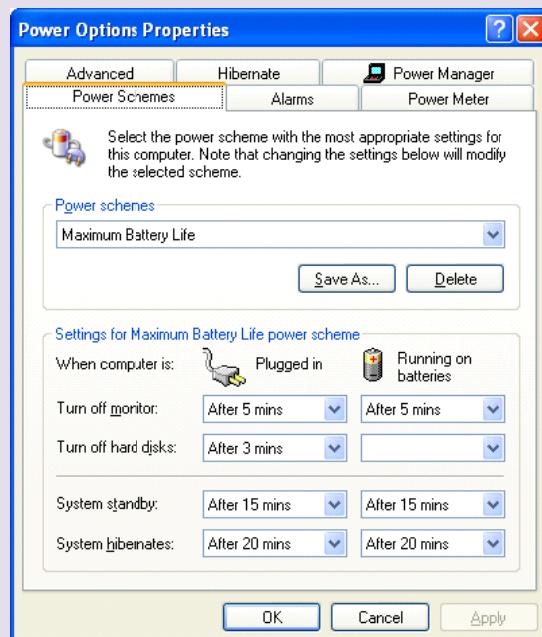
آیا می‌دانید؟

رابط کاربری قسمتی از یک برنامه‌ی نرمافزاری است که کاربر توسط آن با رایانه ارتباط برقرار می‌کند، و به کمک آن ورودی‌های خود را به رایانه داده و خروجی‌های مربوطه را دریافت می‌کند. شکل زیر رابط کاربری برای یک برنامه‌ی نوشته شده در زبان کوئیک بیسیک را نشان می‌دهد.

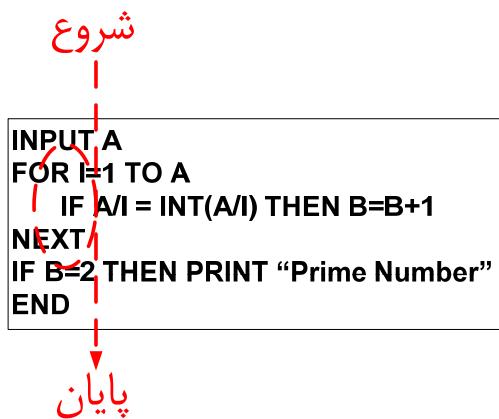


آیا می‌دانید؟

رابط گرافیکی کاربری، در واقع یک رابط کاربری است که در طراحی آن از امکانات گرافیکی مانند تصویر، متن و ... کمک گرفته شده است. اکثر پنجره‌های سیستم‌عامل ویندوز، به نحوی رابط گرافیکی کاربری محسوب می‌شوند. در زیر یک رابط گرافیکی کاربری نشان داده شده است.

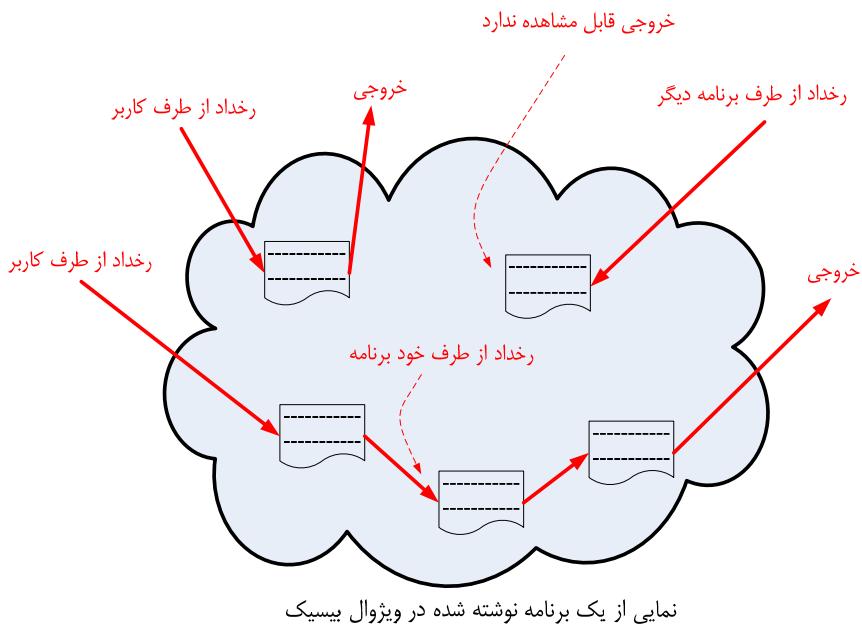


- **مبتنی بر رخداد:** یکی از ویژگی‌های مهم زبان ویژوال بیسیک، مبتنی بر رخداد^۱ (رویداد و یا اتفاق) بودن آن است. اکنون باید دید که این مفهوم به چه معنی است.
اگر یادتان باشد زمانی که در محیط کوئیک بیسیک برنامه نویسی می‌کردید، اجرای برنامه از یک نقطه آغاز و در یک نقطه به پایان می‌رسید و در این بین دستوات نوشته شده توسط شما، با ترتیب خاصی اجرا می‌شدند، که ممکن بود در این میان، بعضی از قسمت‌ها اجرا نشوند و بعضی از قسمت‌ها به واسطه‌ی وجود حلقه‌های تکرار، چند بار اجرا شوند. شکل زیر به نحوی، بیان‌گر این گونه از برنامه‌ها می‌باشد.



اما در ویژوال بیسیک چنین روندی برای اجرای برنامه‌ها وجود ندارد و در واقع برنامه‌طی یک روند مشخص اجرا نمی‌شود، بلکه در پاسخ به رخدادهای مختلف، قسمت‌هایی از برنامه اجرا می‌شود. برای مثال حرکت ماوس، فشار دادن یک دکمه و...، هریک به نحوی یک رخداد محسوب شده و می‌توانند (نه لزوماً) باعث اجرا شدن بخشی از برنامه شوند. باید توجه نمود که رخدادها ممکن است در نتیجه‌ی یک عمل کاربر و یا در اثر پیغام‌هایی باشد که دیگر برنامه‌ها و یا سیستم‌عامل ارسال کرده‌اند. همچنین توجه به این نکته ضروری است که خود برنامه نیز می‌تواند باعث بروز رخداد شود. با این دیدگاه، شکل زیر می‌تواند بیان‌گر این موضوع باشد:

¹ Event



- **تطابق ویژوال بیسیک با دنیای واقعی:** اگر به دنیای اطرافمان نگاهی بیاندازیم، بهوضوح در می‌یابیم که دنیای واقعی از اشیایی (شامل جانداران و ...) تشکیل شده است که در تعامل با یکدیگر هستند. هر یک از این اشیاء دارای خصوصیاتی می‌باشند و در ازای رخدادهایی که بر روی آن‌ها واقع می‌شود، عکس‌عمل‌هایی را از خود نشان می‌دهند. برای واضح‌تر شدن موضوع یک ماشین سواری را در نظر بگیرید. هر ماشین سواری دارای تعدادی خصوصیت است که در واقع نشان‌دهنده و بیان‌گر ماهیت آن است. مانند :

نام	■
پلاک	■
رنگ	■
تعداد درب	■
ظرفیت موتور	■

بر روی هر ماشین سواری رخدادهایی نیز واقع می‌شود:

- فشرده شدن پدال ترمز
- چرخیدن فرمان
- چرخاندن سوییچ در محل مناسب

در ازای رخدادهای مشخص و از پیش تعیین شده‌ای که بر روی ماشین سواری واقع می‌شود، ماشین عکس‌عمل‌هایی نشان می‌دهد:

- متوقف شدن حرکت
- گردش چرخ‌ها
- روشن و خاموش شدن موتور

در دنیای برنامه‌سازی ویژوال بیسیک نیز همین روال وجود دارد. در واقع هر برنامه از تعدادی شیء (که در دنیای ویژوال بیسیک به آن‌ها کترل گفته می‌شود) تشکیل شده است که هر کدام دارای خصوصیاتی هستند و هم‌چنین همان‌طور که در بخش قبل توضیح داده شد، در ازای واقع شدن یک سری رخداد مشخص بر آن‌ها، عکس‌عمل‌هایی از جمله اجرا شدن قطعه برنامه‌ها، به وجود خواهد آمد.

نمونه :

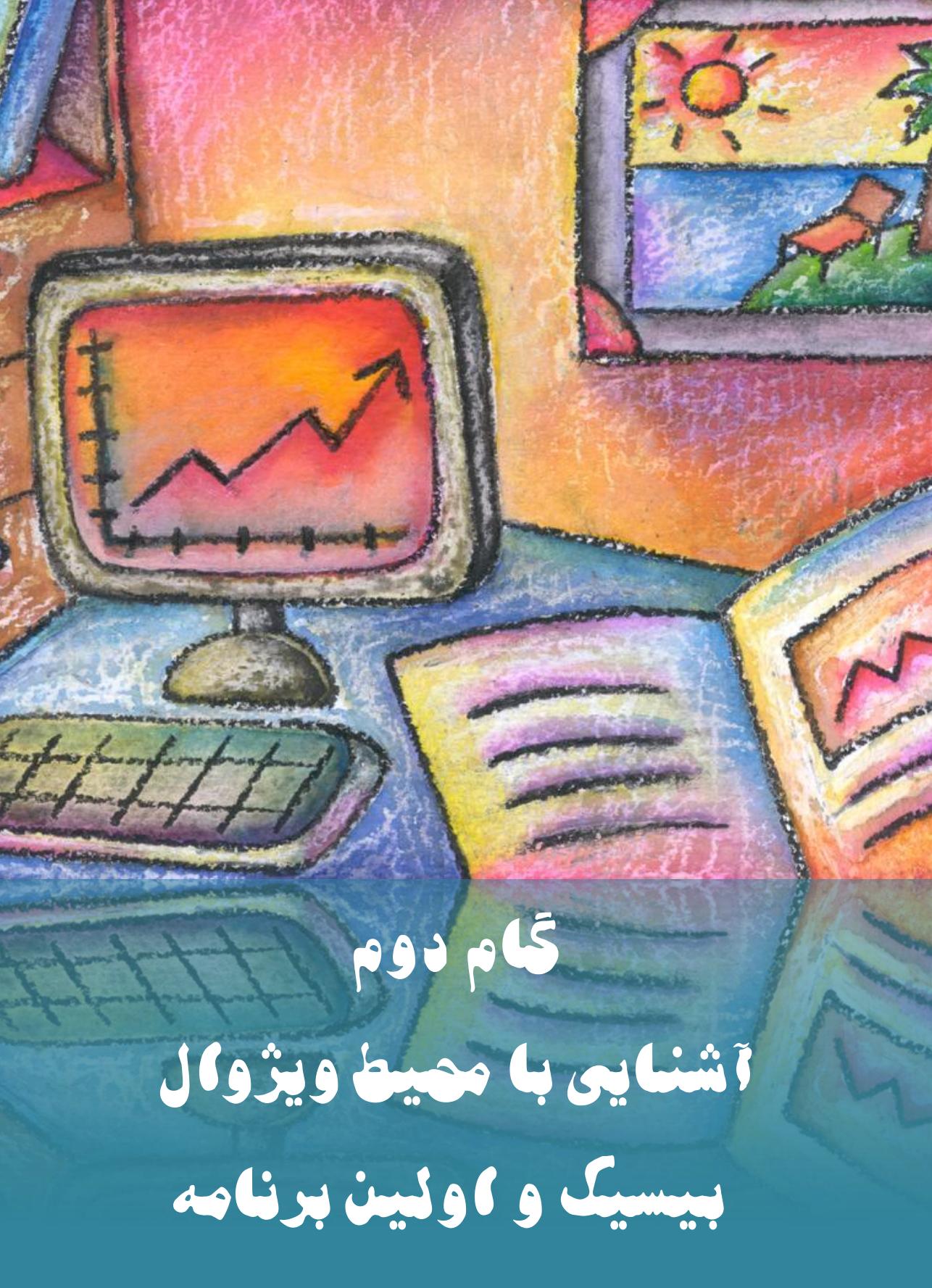
می‌توان یک برنامه برای پخش موسیقی را طوری برنامه‌ریزی کرد که با کلیک کردن بر روی یک دکمه دایره شکل آبی رنگ، موسیقی مورد نظر کاربر پخش شود. باید توجه کرد که این دکمه دارای خصوصیاتی از جمله رنگ آبی است. هم‌چنین در ازای رخداد از پیش مشخص شده‌ی کلیک کردن، موسیقی مورد نظر کاربر، پخش خواهد شد.



۱. ۳. تمرین

۱. چند نمونه از رخدادهای ممکن در دنیای واقعی اطراف خود را ذکر کنید.
۲. هر کدام از این رخدادها بر چه شیئی واقع می‌شود؟
۳. عکس العمل آن شیء در ازای واقع شدن آن رخداد چیست؟
۴. مثال‌هایی از رخدادهای ممکن در محیط ویندوز بیان کنید.





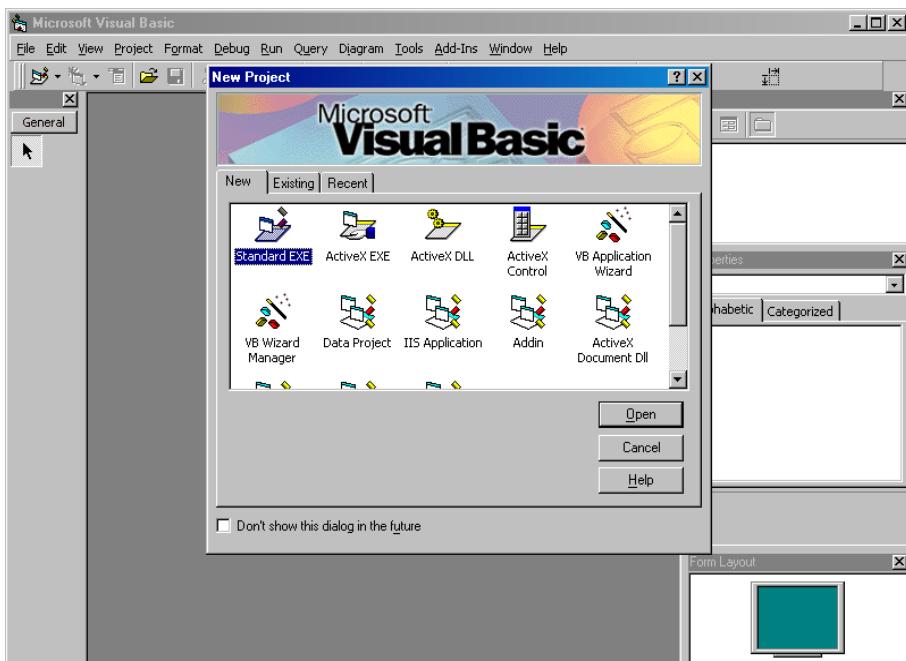
گام دوم

آشنایی با محیط ویژوال بیسیک و اولین برنامه

۱.۱ آشنایی

بعد از کمی آشنایی با فلسفه‌ی اختراع ویژوال بیسیک و همچنین ماهیت برنامه‌های آن، حال نوبت به مشاهده‌ی این محیط رسیده است.

اگر ویژوال بیسیک باز نیست، آن را با دوکلیک (Double Click) بر روی نماد (Icon) برنامه اجرا کنید. ویژوال بیسیک بعد از نصب در منوی شروع ویندوز و در بخش Microsoft Visual Studio 6.0 قرار می‌گیرد. بعد از اجرا، پنجره‌ای مشابه با شکل زیر ظاهر می‌شود. بر روی گزینه‌ی Standard دو کلیک کنید تا پروژه‌ای از نوع معمولی داشته باشیم.



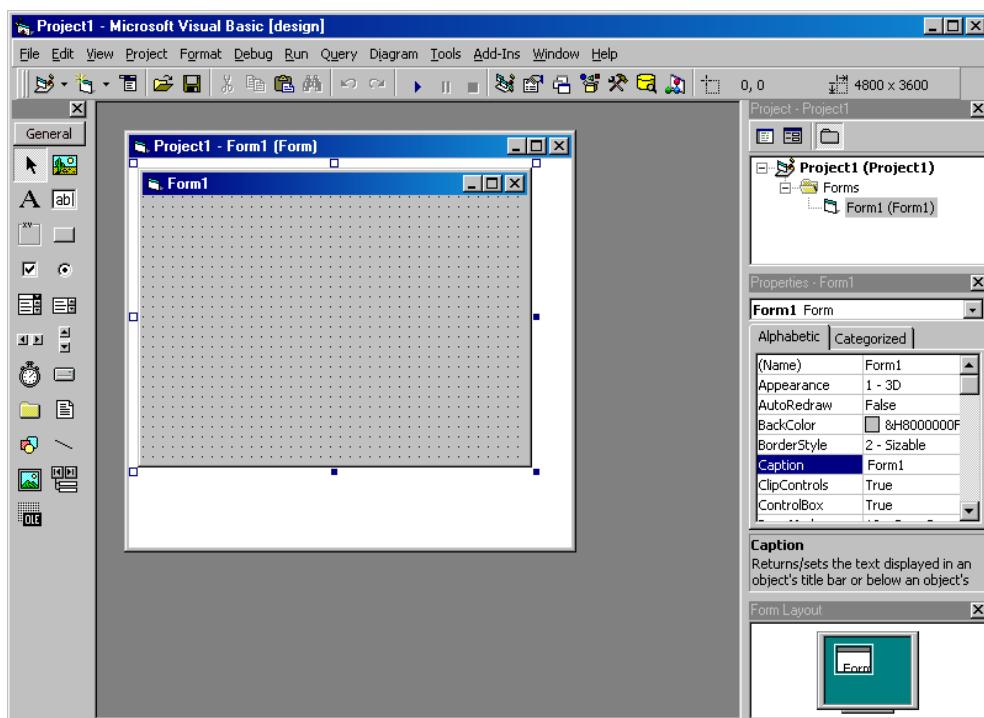
آیا می‌دانید؟

پروژه‌ها در ویژوال بیسیک انواع مختلفی دارند، به طور مثال می‌توانیم پروژه‌ای معمولی داشته باشیم که گزینه‌ی Standard EXE پروژه‌هایی از این دست می‌سازد. بسیاری از برنامه‌های تحت ویندوز از این نوع هستند که می‌توان به برنامه‌هایی چون Paint، WinZip و... اشاره کرد.

از ویژوال بیسیک برای ساختن سایت‌های اینترنتی و اینترانتی نیز می‌توان استفاده کرد. گزینه‌های DHTML Application ، ActiveX Document EXE ، IIS Application می‌روند.

پروژه‌ها انواع دیگری هم دارند که بررسی آن‌ها را به خودتان واگذار می‌کنیم. البته در طی سال ما فقط با پروژه‌های معمولی، یعنی گزینه Standard EXE سر و کار خواهیم داشت.

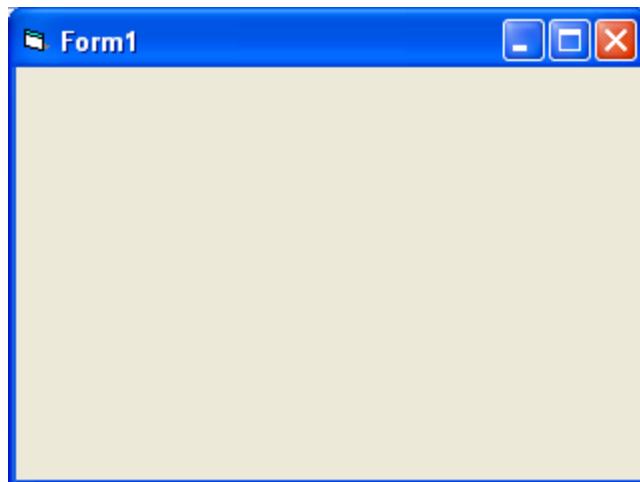
بعد از این که نوع برنامه را Standard EXE برگزیدید، به محیط برنامه‌سازی ویژوال بیسیک می‌رسید. تصویر شکل زیر، محیط برنامه‌سازی ویژوال بیسیک را نمایش می‌دهد. سعی کنید مواردی را که در شکل نشان داده شده است، در محیط ویژوال بیسیک شناسایی کنید.





توجه : شاید متوجه شده باشید که در جلسه‌ی امروز به جای آن که از لغت برنامه‌نویسی استفاده کنیم، از لغت برنامه‌سازی استفاده کرده‌ایم! آیا می‌توانید دلیل آن را حدس بزنید؟ اگر جواب این پرسش را نمی‌دانید، اصلاً جای نگرانی نیست! تا انتهای این جلسه، جواب این پرسش برایتان بدیهی خواهد بود!

قبل از هر چیز بد نیست برنامه‌ای را که ننوشته‌ایم^(۱) اجرا کنیم و ببینیم که نتیجه چیست؟ کلید F5 را بزنید. یک پنجره (frm^۱) مانند شکل زیر می‌بینید که بر روی آن هیچ چیزی نیست. از این به بعد ظاهر برنامه‌هایمان را بر روی فرم خواهیم ساخت.



انتظار نداشته باشید که برنامه به خودی خود به اتمام رسد، زیرا باید خودتان این کار را انجام دهید. با کلیک کردن بر روی علامت X که در گوشه‌ی بالا و سمت راست فرم قرار دارد، اجرای برنامه را خاتمه دهید تا به محیط برنامه‌سازی بازگردید.

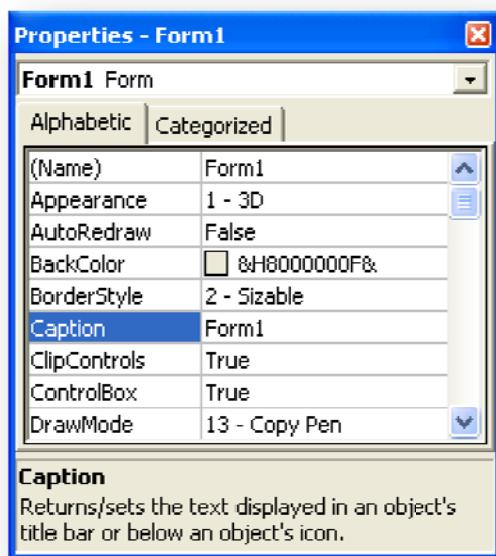
^۱ Form

۲. تغییر عنوان فرم

در اولین گام می‌خواهیم خصوصیتِ عنوان (Caption) فرمی را که در اختیار داریم، تغییر دهیم. زمانی که پروژه را آغاز کردید، یک فرم خالی ایجاد شد که دارای عنوان Form1 بود. در واقع این عنوانی است که به صورت پیش‌فرض، برای این فرم اختصاص داده شده است. هر چند تغییر ندادن این عنوان، مشکلی را ایجاد نمی‌کند، ولی بهتر است که عنوان مناسب و مورد پسندی را انتخاب کنیم. به دلیل این که امروز اولین برنامه را در محیط ویژوال بیسیک ایجاد می‌کنیم، عنوان فرم را به My First Program تغییر خواهیم داد. برای این کار، لازم است کارهای زیر را انجام دهیم:

۱. قبل از هر چیز باید مطمئن باشید که فرم انتخاب شده است. برای انتخاب کردن فرم کافی است بر روی آن یک کلیک انجام دهید.

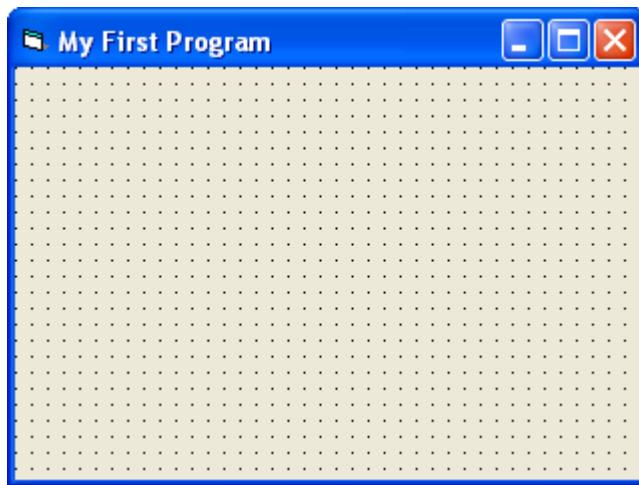
۲. در این مرحله اگر پنجره خصوصیات (Properties) باز نیست، با انتخاب گزینه View از منوی Properties Window آن را ظاهر نمایید. این پنجره، ظاهراً به صورت زیر دارد:



۳. بر روی خانه‌ای که در سمت راست آن کلمه **Caption** نوشته شده است، کلیک نمایید (به معنی عنوان است).

۴. در خانه‌ی مقابل آن عبارت **Form1** دیده می‌شود که همان عنوان فرم می‌باشد. با کلیک کردن بر روی آن، محتوای آن را به عنوان مورد نظر یعنی **My First Program** تغییر دهید.

شما موفق به تغییر عنوان فرم شده‌اید. اگر نگاهی به فرم بیاندازید متوجه خواهید شد!

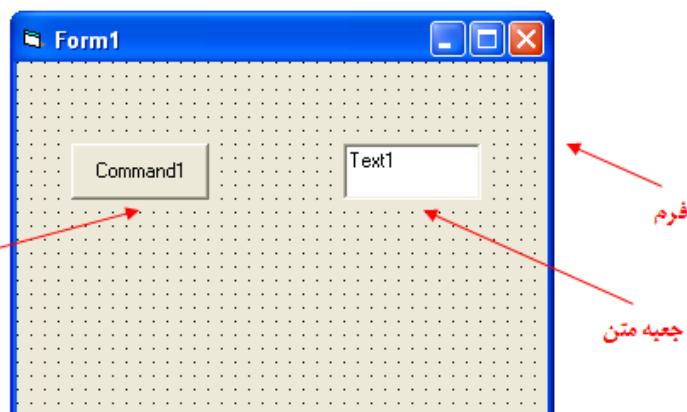


شاید بد نیاشد که قبل از ادامه کار ببینیم که **Property** یا همان خصوصیت چیست و پنجره‌ی خصوصیات به چه کار می‌آید؟

(Property) ۳. خصوصیت

باید توجه کرد که عنوان و یا همان **Caption**، یکی از خواص فرم است و اگر با دقت به پنجره‌ی خصوصیات نگاه کنید، تعداد زیادی خصوصیت در آن به چشم می‌خورد. این خصوصیات تعیین‌کننده ظاهر و بعضی از رفتارهای اشیاء و کنترل‌هایی است که در محیط ویژوال بیسیک، مورد استفاده قرار می‌گیرند. اگر یادتان باشد در جلسه قبل گفتم که هر یک از اشیاء و یا کنترل‌ها، دارای تعدادی خصوصیت هستند. در محیط برنامه‌سازی ویژوال بیسیک نیز کنترل‌های گوناگونی وجود دارند که هر

یک دارای خصوصیاتی هستند که در پنجره خصوصیات بیان می‌شوند. از این کنترل‌ها می‌توان به فرم، دکمه فرمان، جعبه متن و... اشاره کرد. برای استفاده از یک کنترل، از جعبه ابزار استفاده خواهیم کرد که در ادامه توضیح داده خواهد شد.



۴. آشنایی با جعبه‌ی ابزار

جعبه ابزار پنجره‌ای است مشابه شکل روبرو که شامل تعدادی نماد (Icon) برای کنترل‌های مختلف قابل استفاده، در ایجاد یک برنامه می‌باشد. در واقع شما برای استفاده از این کنترل‌ها، باید آن‌ها را از جعبه ابزار انتخاب کرده و در محل مناسبی از فرم قرار دهید.

توجه



در صورتی که در محیط کار شما، جعبه ابزار دیده نمی‌شود می‌توانید با انتخاب گزینه Toolbox از منوی View آن را ظاهر نمایید. در ضمن توجه به این نکته ضروری است که با توجه به محل قرارگیری جعبه ابزار در محیط ویژوال بیسیک، اندازه‌ی این جعبه می‌تواند متفاوت از آن چیزی باشد که در شکل قبل نشان داده شده است.

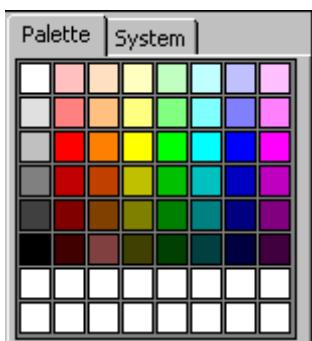
نکته



در صورتی که نشان‌گر ماوس را بر روی هر یک از کنترل‌های داخل جعبه ابزار نگه دارید (بدون کلیک کردن)، مربع زرد رنگی ظاهر می‌شود که نام کنترل مورد اشاره را بیان می‌کند.

۲.۵. اولین برنامه در ویژوال بیسیک

می‌خواهیم برنامه‌ای بسازیم که رمز عبوری را از کاربر پرسیده و اگر رمز درست بود، پیغام «شما می‌توانید وارد شوید» و اگر رمز نادرست بود، پیغام «شما نمی‌توانید وارد شوید» را بدهد. می‌خواهیم فرم این برنامه، دارای یک دکمه‌ی فرمان برای تایید و یک دکمه‌ی فرمان نیز برای خاتمه‌ی برنامه باشد. در ادامه، به بررسی مراحل کار می‌پردازیم.



کار را با تغییر بعضی از خصوصیت‌های فرم آغاز می‌کنیم. توجه داشته باشید که در مراحل قبلی خصوصیت Caption فرم را تغییر داده‌ایم. اکنون بر روی خصوصیت BackColor دوکلیک کرده تا جعبه رنگ، مطابق شکل رو به رو، باز شود. این جعبه دارای دو برگه است. برگه Palette را انتخاب نموده و سپس از داخل آن، رنگ دلخواه خود را برگزینید.

از جعبه ابزار، یک کنترل از نوع **Label** (برچسب)، روی فرم بگذارید. این کار را می‌توان با دو کلیک کردن بر روی نماد **Label** در جعبه ابزار، انجام داد. مشاهده می‌کنید که یک **Label** روی فرم قرار می‌گیرد. معمولاً از کنترل **Label** برای نمایش متن، استفاده خواهیم کرد. خصوصیات **Label** را به مقادیر خواسته شده، تغییر دهید. توجه کنید همان‌طور که قبل نیز توضیح داده شد، قبل از تغییر خصوصیات یک کنترل، ابتدا باید آن را با ماوس انتخاب کرده باشید.

نام	مقدار
Name	Lbl_Password
Caption	Password
BackColor	قرمز
ForeColor	آبی
Font	Size = 14
Left	300
Top	300

از جعبه ابزار، یک کنترل **TextBox** (جعبه‌ی متن) روی فرم بگذارید. از **TextBox** برای گرفتن متن و یا عدد استفاده می‌کنیم. (توجه کنید که در ویژوال بیسیک، برخلاف بیسیک دیگر دستور **Input** نداریم ولی به جای آن می‌توان از امکانات بهتری از جمله جعبه‌ی متن استفاده نمود) خصوصیات **TextBox** را به مقادیر خواسته شده تغییر دهید:

نام	مقدار
Name	Txt_Password
Text	
BackColor	قرمز
ForeColor	آبی
Font	Size = 14
Left	2400
Top	300
Width	3000

از جعبه ابزار، یک کنترل **CommandButton** (دکمه‌ی فرمان) روی فرم بگذارید. خصوصیات آن را به مقادیر خواسته شده تغییر دهید:

نام	مقدار
Name	Cmd_OK
Caption	OK
Left	1000
Top	1500

از جعبه ابزار، یک کنترل **CommandButton** دیگر روی فرم بگذارید. خصوصیات آن را به مقادیر خواسته شده تغییر دهید:

نام	مقدار
Name	Cmd_Exit
Caption	Exit
Left	1000
Top	3500

برنامه را با زدن کلید **F5** اجرا نمایید. در جعبه متن چیزی بنویسید و بر روی دکمه‌های OK و Exit کلیک کنید. آیا برنامه‌ی شما کاری انجام می‌دهد؟ چرا؟
بله، درست است. زیرا شما تا این لحظه هیچ برنامه‌ای نوشته‌اید و تنها ظاهر (ویژوال) برنامه را ساخته‌اید. حال بهتر است برنامه‌نویسی را آغاز کنیم.
با ماوس روی دکمه Exit دو کلیک کنید تا پنجره‌ی برنامه‌نویسی مربوط به آن، باز شود. عبارت زیر، از قبل نوشته شده است:

```
Private Sub Cmd_Exit_Click()
```

```
End Sub
```

بین دو خط موجود، دستور **End** را بنویسید. همانند کوئیک بیسیک، این دستور باعث خاتمه‌ی کل برنامه خواهد شد. برنامه‌ای که دارد، اکنون به صورت زیر در آمده است.

```
Private Sub Cmd_Exit_Click()
    End
End Sub
```

برنامه را با **F5** اجرا کنید. با ماوس روی دکمه Exit کلیک کنید. چه اتفاقی رخ می‌دهد؟ شما چه انتظاری داشتید؟ همان‌طور که گفتیم، دستور End باعث خاتمه یافتن برنامه می‌شود. بیایید برنامه را کامل کنیم. روی دکمه OK دو کلیک کنید تا پنجره‌ی برنامه نویسی مربوط به آن باز شود. عبارت زیر، از قبل نوشته شده است:

```
Private Sub Cmd_Ok_Click()
```

```
End Sub
```

در بین دو خط موجود، کد (Code) و یا قطعه برنامه زیر را اضافه کنید. (کد در لغت به معنی دستور و در عمل یک تکه برنامه است).

```
If Txt_Password.Text = "Sampad" Then
    MsgBox " You Can Come In ! "
Else
    MsgBox " You Cannot Come In ! "
End If
```

آیا می‌دانید؟

MessageBox مخفف Message Box بوده (به معنای جعبه‌ی پیغام) و دستوری است که یک رشتہ را گرفته و آن را در یک پنجره نمایش می‌دهد. در زیر نمونه‌هایی از جعبه پیغام، نشان داده شده است.



برنامه را با **F5** اجرا کرده و نتیجه کارتان را مشاهده کنید. یکبار در جعبه‌ی متن، کلمه Sampad را وارد و دکمه OK را بزنید و بار دیگر کلمه‌ای به‌غیر از Sampad وارد کرده و دکمه OK را بزنید و نتیجه را ببینید.

تمرین

چه کار کنیم تا وقتی که می‌خواهیم Password را وارد کنیم، فقط * نمایش داده شود (تا دیگران متوجه نشوند ما چه چیزی تایپ کردہ‌ایم)؟
 (راهنمایی : یکی از خصوصیات جعبه‌ی متن (TextBox)، به این قضیه مربوط می‌شود. آیا می‌توانید آن را پیدا کنید؟)

نکته

بله، به نظر می‌رسد که درست حدس زده‌اید!
 تولید برنامه در زبان ویژوال بیسیک، شامل دو مرحله‌ی جداگانه می‌باشد:

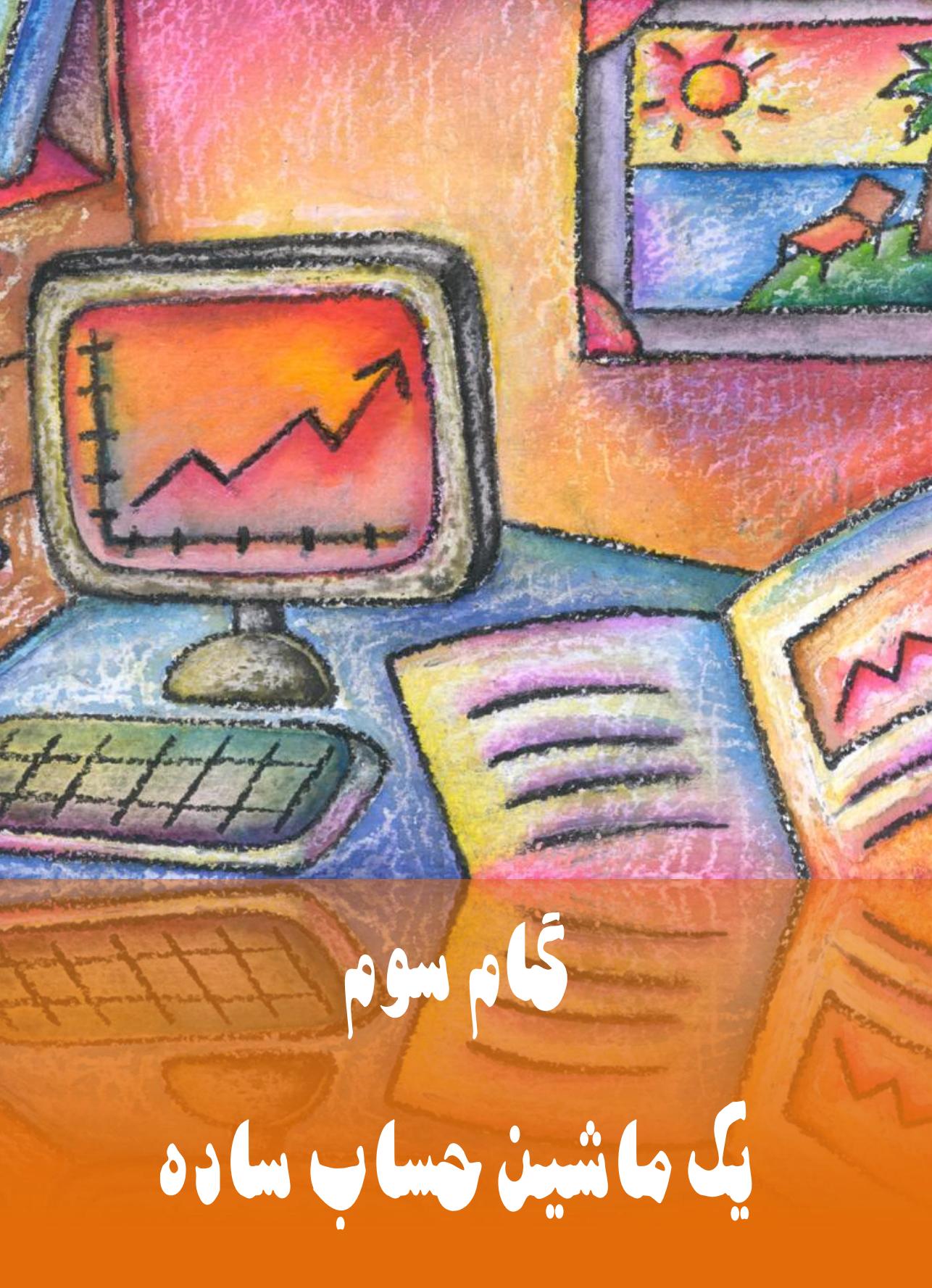
- تولید رابطه‌ای کاربر و به عبارت دیگر بخش گرافیکی (ویژوال!)
- تولید برنامه‌های لازم و به عبارت دیگر برنامه نویسی (بیسیک!)

در نتیجه، تولید برنامه‌ها در ویژوال بیسیک، تنها به برنامه‌نویسی ختم نمی‌شود و ایجاد رابطه‌ای کاربر و بخش گرافیکی نیز قسمت مهمی از فرآیند تولید برنامه را تشکیل می‌دهد.



یک ماشین حساب ساده

گام سوم



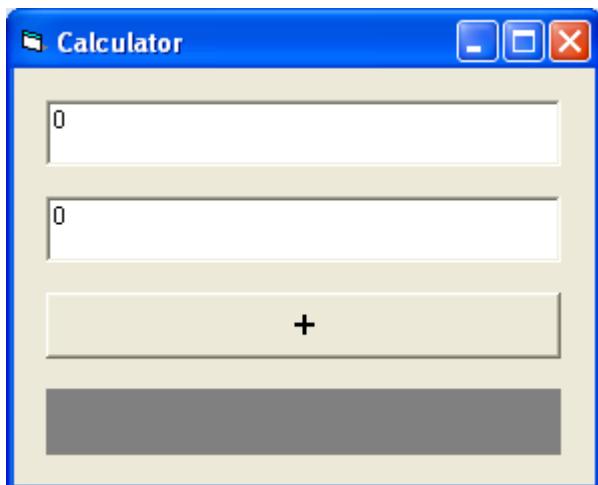
۳.۱ آشنایی

تا اینجا دیدید که ویژوال بیسیک هم مانند دیگر زبان‌ها، زبانی برای تولید برنامه‌های کامپیوتراست که البته دارای امکانات گسترده‌ای برای هر چه بهتر کردن برنامه‌هاست.

در این جلسه، به منظور آشنایی بیشتر با ویژوال بیسیک، یک ماشین حساب ساده را طراحی کرده و در این میان با محیط ویژوال بیسیک نیز آشنایی بیشتری پیدا خواهیم کرد. پس از اجرا کردن ویژوال بیسیک، یک پروژه از نوع Standard EXE را ایجاد نمایید.

۳.۲ برنامه‌ی جمع دو عدد

ابتدا بهتر است نگاهی به نتیجه نهایی فرم بیاندازیم و بدانیم که در طی مراحل آتی، به دنبال چه هدفی هستیم.



با وارد کردن دو عدد و زدن دکمه جمع، حاصل جمع دو عدد وارد شده باید در برچسبی که پررنگ‌تر از بقیه مشخص شده است، نشان داده شود. با دنبال کردن مراحل زیر، می‌توانید به هدف مذکور برسید.

همان‌طور که در شکل فوق نیز دیده می‌شود، برای گرفتن هر عدد از یک جعبه‌ی متن (TextBox) جداگانه و برای نمایش حاصل جمع از یک برچسب (Label) استفاده می‌کنیم.

۳. چگونه کار خود را ذخیره کنیم؟

اما قبل از آنکه کار را شروع کنیم، می‌خواهیم پروژه‌ای را که آغاز کرده‌ایم، ذخیره (Save) نماییم. درست است که تغییر زیادی را ایجاد نکرده‌اید، اما بهتر است عادت کنید که کارهای خود را گام به گام ذخیره نمایید.

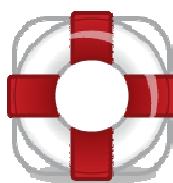
برای ذخیره کردن برنامه باید دو نوع فایل را ذخیره کنید:

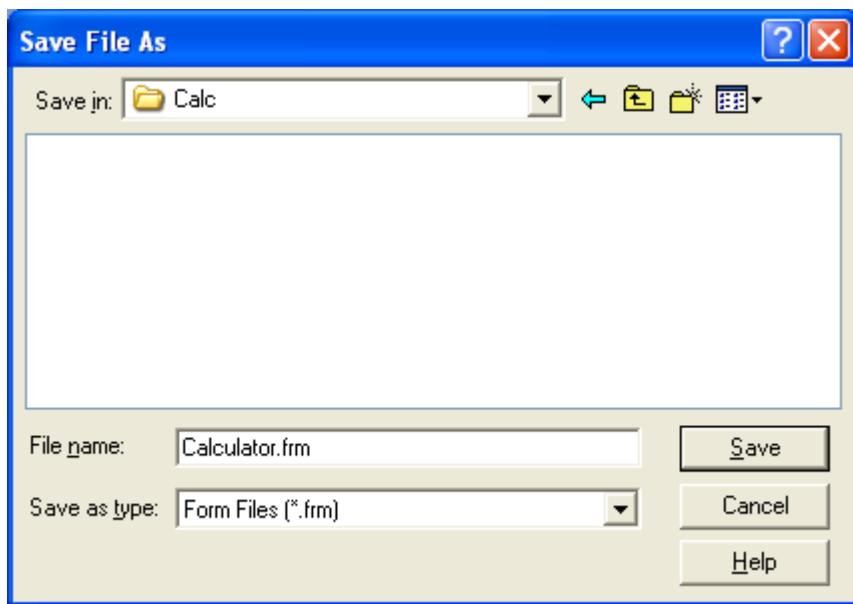
- فایل مربوط به پروژه، که دارای پسوند **.vbp** است و حاوی اطلاعاتی است که برای ایجاد پروژه استفاده می‌شود.
- فایل مربوط به فرم، که دارای پسوند **.frm** است و حاوی اطلاعات مربوط به فرم می‌باشد.

برای ذخیره کردن کارتان، مراحل زیر را دنبال کنید:

۱. ابتدا مطمئن باشید که فرم را انتخاب کرده‌اید و عنوان آن پر رنگ باشد. (برای اطمینان کافی است یک کلیک بر روی آن انجام دهید). اکنون گرینه‌ی **Save Form1 As** پنجره‌ای باز شود. فرم را در محلی که می‌خواهید، ذخیره نمایید. بهتر است برای این که نظمی به کارتان داده باشید، پوشه‌ای برای این پروژه ایجاد نموده و تمامی فایل‌های مربوط به آن را در همان پوشه ذخیره کنید.

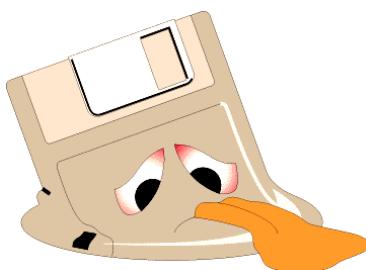
۲. زمانی که پنجره ذخیره کردن باز می‌شود، نام **Form1** را پیشنهاد می‌کند. ترجیحاً از این نام استفاده نکنید تا در آینده به راحتی از روی نام فایل، بتوانید کارتان را تشخیص دهید. برای مثال **Calculator.frm** می‌تواند نام مناسبی باشد.





۳. در گام بعدی، باید فایل مربوط به پروژه را ذخیره نمایید. گزینه **Save Project As** را از منوی **File** انتخاب نموده و مشابه مرحله‌ی قبل، فایل مربوط به پروژه را در همان پوشه‌ای که ساخته‌اید، ذخیره نمایید. باز تاکید می‌کنیم که از نام‌های پیش فرضی چون **Project1** استفاده نکنید. نام پروژه را خودتان تعیین کنید. برای مثال **Calculator.vbp** عنوان مناسبی به نظر می‌رسد.

اکنون شما کارتان را با موفقیت ذخیره کرده‌اید. حالا نوبت به طراحی فرم با استفاده از کنترل‌های جعبه ابزار رسیده است.



۴.۳. طراحی پروژه

ابتدا خواص فرم را مطابق با جدول زیر تغییر دهید:

نام	مقدار
Name	Frm_Calculator
Caption	Calculator
Height	3600
Width	4500

خصوصیت Name این فرم، برابر Frm_Calculator قرار داده شده است و با این نام در برنامه شناخته می‌شود. ویژوال بیسیک به صورت پیش‌فرض، آن را برابر From1 قرار داده بود. توجه کنید که این (Frm_Calculator) نامی است که در محیط ویژوال بیسیک شناخته شده است و نباید آن را با Calculator.frm که در مرحله‌ی قبل، فرم را به آن نام ذخیره کردید، اشتباه کنید. نام فایلی Calculator.frm است که اطلاعات مربوط به فرم، در آن نگهداری می‌شود.

یک جعبه‌ی متن بر روی فرم برنامه بگذارد و خواص آن را مطابق جدول زیر تغییر دهید:

نام	مقدار
Name	Txt_No1
Text	0
Height	495
Width	3900
Left	240
Top	240

توجه داشته باشید که خصوصیت Name این جعبه متن، Txt_No1 قرار داده شده است و با این نام در برنامه شناخته می‌شود. ویژوال بیسیک به صورت پیش‌فرض، آن را برابر Text1 قرار داده بود.

یک جعبه‌ی متن دیگر، بر روی فرم برنامه بگذارید و خواص آن را مطابق جدول زیر تغییر دهید:

نام	مقدار
Name	Txt_No2
Text	0
Height	495
Width	3900
Left	240
Top	960

توجه داشته باشید که خصوصیت Name این جعبه متن، Txt_No2 می‌باشد و با این نام در برنامه شناخته می‌شود.

یک دکمه‌ی فرمان (CommandButton) نیز بر روی فرم برنامه بگذارید و خواص آن را مطابق زیر تغییر دهید:

نام	مقدار
Name	Cmd_Sum
Caption	+
Font	Size = 14
Height	495
Width	3900
Left	240
Top	1680

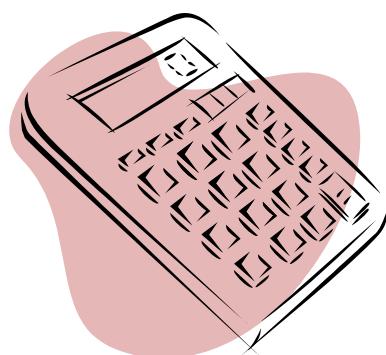
توجه داشته باشید که خصوصیت Name این دکمه فرمان Cmd_Sum می‌باشد و با این نام در برنامه شناخته می‌شود.

و در نهایت یک Label بر روی فرم برنامه بگذارید و خواص آن را مطابق جدول صفحه بعد تغییر دهید:

نام	مقدار
Name	Lbl_Result
BackColor	خاکستری
Caption	
Font	Size = 14
Height	495
Width	3900
Left	240
Top	2400

توجه داشته باشید که خصوصیت Name این برچسب، Lbl_Result می‌باشد و با این نام در برنامه شناخته می‌شود.

تاکنون فقط ظاهر برنامه را درست کرده‌ایم. اکنون برنامه را با **F5** اجرا کرده تا حاصل کار خود را ببینید. ظاهر برنامه باید مطابق با شکل اول باشد. اکنون اجرای برنامه را متوقف کرده تا پس از بررسی بعضی نکات مهم، به برنامه‌نویسی پردازیم.



 آیا می‌دانید؟

همان طور که تاکنون مشاهده کرده‌اید، در طی تنظیم خصوصیات کنترل‌های، به کار رفته در برنامه، نام آن‌ها (خصوصیت Name) را نیز تغییر داده‌ایم. خصوصیت Name یک کنترل، در واقع بیان‌گر نام آن بوده و آن کنترل در طی برنامه با آن نام شناخته می‌شود. باید توجه کرد که نام دو کنترل متمایز، نمی‌تواند یکسان باشد. در ضمن برای واضح‌تر بودن نام کنترل‌ها، از استاندارد زیر استفاده خواهیم کرد. ابتدای نام هریک از کنترل‌ها، طبق جدول زیر، یک کلمه اضافه می‌کنیم. این کار باعث خواهد شد تا با مشاهده‌ی نام کنترل، به نوع آن پی ببریم و در نتیجه کارمان در زمان برنامه‌نویسی بسیار ساده‌تر و اصولی‌تر خواهد بود.

نوع کنترل	۴ حرف ابتدایی نام کنترل	مثال
Form	Frm_	Frm_Main
Check Box	Chk_	Chk_Color
Combo Box	Cmb_	Cmb_Field
Command Button	Cmd_	Cmd_OK
Frame	Fra_	Frame_Choices
Horizontal Scroll Bar	Hsb_	Hsb_Position
Image	Img_	Img_Company
Label	Lbl_	Lbl_Name
List Box	Lst_	Lst_Books
Menu	Mnu_	Mnu_View
Option Button	Opt_	Opt_Check
Picture Box	Pic_	Pic_Logo
Text Box	Txt_	Text_Number
Timer	Tmr_	Tmr_Move
Vertical Scroll Bar	Vsb_	Vsb_Speed

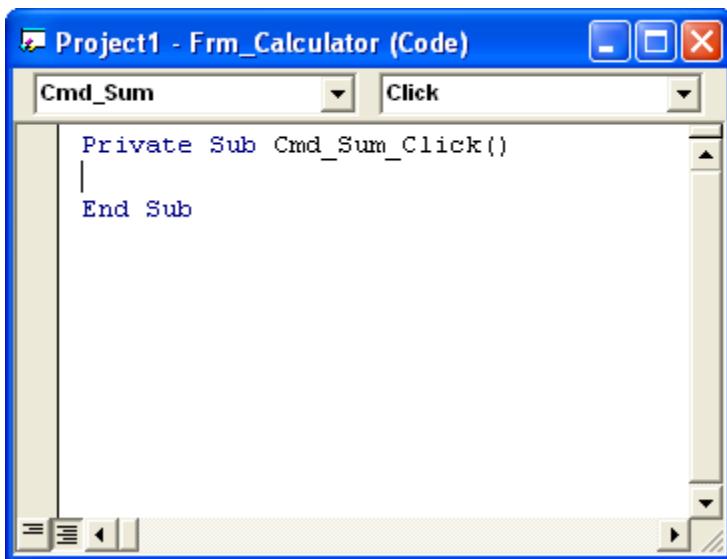
سعی کنید نه تنها در زمان انجام تمرینات این کتاب، بلکه حتی در برنامه‌هایی که خودتان می‌نویسید، از چنین استانداردهایی برای نام‌گذاری کنترل‌ها، استفاده نمایید. در صورتی که از کنترلی، خارج از لیست بالا، استفاده می‌کنید، می‌توانید از روشی مشابه (به کمک مخفف کردن نوع آن) استفاده کنید.

بر روی دکمه‌ی فرمان، دو کلیک کنید تا پنجره‌ی برنامه نویسی مربوط به آن، مطابق با شکل زیر، باز شود. عبارتی که اکنون می‌بینید، تعریف یک زیر برنامه است (زیر برنامه قسمتی از یک برنامه است). این زیر برنامه وقتی اجرا می‌شود که کاربر با ماوس روی دکمه فرمان Cmd_Sum کلیک کند.

این مطلب از دو جعبه‌ای که در بالای این پنجره وجود دارد، مشخص است.

- متن موجود در جعبه‌ی سمت چپ Cmd_Sum بوده که در واقع نام دکمه‌ی فرمان است.
- متن موجود در جعبه‌ی سمت راست Click بوده که بیان گر رخداد کلیک شدن است.

دو مطلب فوق بدین معنی است که این زیر برنامه زمانی اجرا خواهد شد که رخداد کلیک (مقدار جعبه‌ی سمت راست) بر کنترل مورد نظر، یعنی دکمه‌ی فرمان با نام Cmd_Sum (مقدار جعبه‌ی سمت چپ) واقع شود. این موضوع از نام زیر برنامه نیز مشخص است: Cmd_Sum_Click.



اکنون نوبت به برنامه‌نویسی رسیده است. روند کارهایی که می‌خواهیم انجام شود، بدین صورت است:

1. یک عدد در Txt_No1 وارد کنیم. (این عدد در خاصیت Text ذخیره می‌شود.)

.۲. یک عدد دیگر در `Txt_No2` وارد کنیم. (این عدد نیز در خاصیت `Text` ذخیره می‌شود). باید توجه کنید، هر متнی که در یک جعبه متن قرار می‌گیرد، در خاصیت `Text` آن ذخیره می‌شود.

.۳. با کلیک کردن بر روی `Cmd_Sum`، حاصل جمع دو عدد وارد شده، در `Lbl_Result` نمایش داده می‌شود.

کد زیر را در پنجره‌ی کد و بین دو عبارت موجود بنویسید و برنامه را اجرا کنید:

```
Lbl_Result.Caption = Val ( Txt_No1.Text ) + Val ( Txt_No2.Text )
```

سعی کنید نحوه‌ی عملکرد این کد را کشف کنید.

پرسش

اگر در برنامه‌ی بالا از دستور `Val` استفاده نکنیم، چه اتفاقی می‌افتد؟

نکته

اگر به قطعه برنامه‌ی بالا دقت کنید، می‌بینید که در سه جا از علامت نقطه استفاده شده است!
می‌خواهیم ببینیم این علامت به چه معنی است؟

`Lbl_Result • Caption`

`Txt_No1 • Text`

`Txt_No2 • Text`

همان‌طور که تاکنون متوجه شده‌اید، هر کنترل مانند جعبه‌ی متن (Text Box)، دکمه‌ی فرمان (Command Button)، برچسب (Label) و ...، دارای تعدادی خصوصیت (Property) و متد (Method) می‌باشد (در جلسات بعدی با متدها نیز آشنا خواهید شد). برای مثال همه‌ی جعبه‌ی متن‌ها، دارای خصوصیتی به نام `Text` هستند و هر عبارتی که در جعبه‌ی متن قرار گیرد، در واقع در خصوصیت `Text` آن قرار می‌گیرد.

برای تشخیص این که یک خصوصیت و یا متد مربوط به چه کنترلی است، از علامت نقطه استفاده می‌کنیم.

به مثال‌های زیر توجه کنید:

Lbl_Result • Caption

يعني خصوصيت Caption مربوط به كنترلي با نام Lbl_Result (براي اين كه عبارتی را در يك برچسب نشان دهيم، باید آن را در خصوصيت Caption مربوط به آن برچسب قرار دهيم).

Txt_No1 • Text

يعني خصوصيت Text مربوط به كنترلي به نام .Txt_No1

Txt_No2 • Text

يعني خصوصيت Text مربوط به كنترلي به نام .Txt_No2

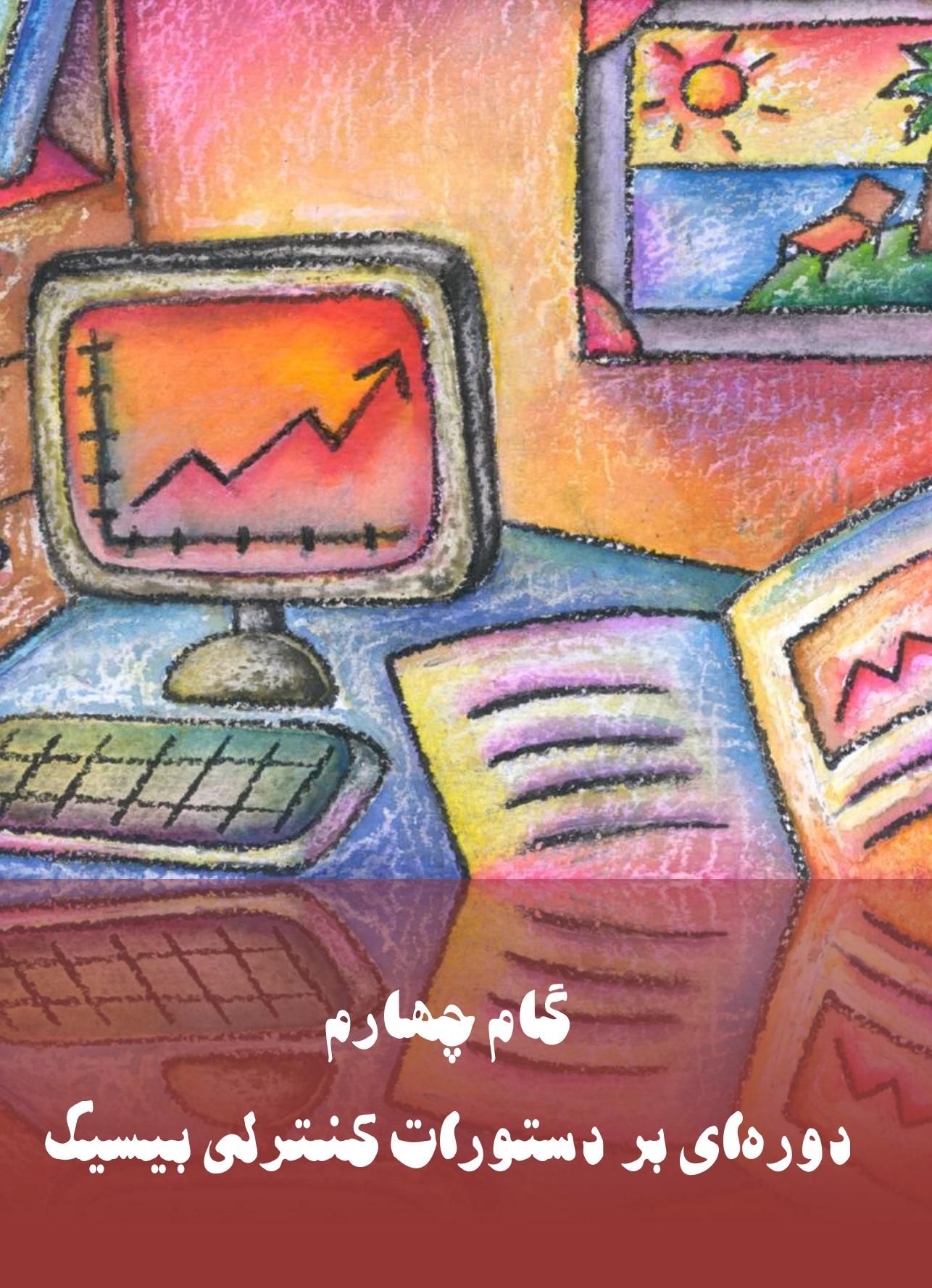
تمرین

برنامه فوق را به نحوی کامل کنید که امکانات تفریق، ضرب و تقسیم را نیز داشته باشد.



گام چهارم

دوره‌ای بر دستورات کنترلی بیسیک



۴. ۱. آشنایی

در جلسه امروز بر دستورات کنترلی زبان بیسیک که در سال گذشته با آن‌ها آشنا شدید، مروری خواهیم داشت.

۴. ۲. ساختارهای شرطی

با استفاده از ساختارهای شرطی می‌توان برای اجرا شدن یک و یا چند دستور شرطهایی را قرار داد.

If ... Then . ۱ . ۲ . ۴

این ساختار را می‌توان به یکی از دو صورت زیر استفاده نمود:

If condition Then statements

If condition Then ←

statements

End If

توجه: در این حالت، عبارتی در مقابل Then قرار نمی‌گیرد و دستوراتی که در صورت برقرار بودن شرط باید اجرا شوند را در خطهای بعدی قرار خواهیم داد.

در صورتی که شرط (condition) برقرار باشد، مجموعه دستورات پس از Then (یعنی statement) اجرا خواهد شد.

پیشنهاد می‌کنیم زمانی که قرار است در مقابل Then چندین دستور و عبارت را قرار دهید از ساختار دوم استفاده نموده تا خوانایی برنامه حفظ شود.

نمونه : 

```
If Int(A / 2) = A / 2 Then MsgBox "Even Number!"
```

```
If Int(A / 2) = A / 2 Then
    MsgBox "Even Number!"
End If
```

If ... Then ... Else .۲ .۲ .۴

این ساختار نیز یک ساختار شرطی محسوب می‌شود که در صورت برقرار بودن و یا نبودن شرط، باعث اجرا شدن دستوراتی می‌شود. کلی ترین حالت این ساختار به صورت زیر می‌باشد:

حالت ساده:

```
If condition Then
    statements-1
Else
    statements-2
End If
```

در صورتی که شرط condition برقرار باشد، مجموعه دستورات statements-1 اجرا خواهد شد و در غیر این صورت (شرط برقرار نباشد) مجموعه دستورات statements-2 اجرا خواهد شد.

 نمونه :

```
If Int(A / 2) = A / 2 Then
    MsgBox "Even Number!"
Else
    MsgBox "Odd Number!"
End If
```

این ساختار شرطی به صورت تودرتو نیز قابل استفاده می‌باشد. برای روشن‌تر شدن بحث به ساختار و روش استفاده آن در ادامه توجه کنید:

حالت تودرتو:

```
If condition1 Then
    statements-1
ElseIf condition2 Then
    statements-2
ElseIf condition3 Then
    statements-3
...
Else
    statements-n
End If
```

در صورتی که شرط condition1 برقرار باشد، مجموعه دستوات statements-1 اجرا خواهد شد. در صورتی که شرط condition1 برقرار نباشد و شرط condition2 برقرار باشد، مجموعه دستوات statements-2 اجرا خواهد شد. در صورتی که شرطهای condition1 و condition2 برقرار نباشند و شرط condition3 برقرار باشد، مجموعه دستورات statement-3 اجرا خواهد شد و ... و در نهایت در صورتی که هیچ‌یک از شرطها برقرار نباشند، مجموعه دستورات statements-n اجرا خواهند شد.

توجه کنید که وجود هریک از بخش‌های رنگی (قرمز، آبی، ... و سبز) در ساختار بالا اختیاری است.

 نمونه :

```
If Number < 10 Then
    MsgBox "1 Digit Number"
ElseIf Number < 100 Then
    MsgBox "2 Digit Number"
Else
    MsgBox "A Number with More Than 2 Digits"
End If
```

ساختار شرطی ... Case Select

اين ساختار نيز يك ساختار شرطی است که عملكردي مشابه با ساختار ... If ... Then ... Else ... Elself ... دارد. قبل از توضيح بيشتر بهتر است نگاهي بر ساختار آن داشته باشيم:

Select Case expression

Case expression-list1

statements-1

Case expression-list2

statements-2

...

Case Else

statements-n

End Select

در ابتداي کار مقدار **expression** محاسبه شده و در صورتی که با از اقلام **expression-list** ها برابر شود، دستورات مربوط به آن بخش اجرا خواهد شد و اگر با هيچ يك از اقلام برابر نشود، دستورات بخش **Else** اجر خواهد شد.

به اين نكته‌ی مهم توجه داشته باشيد که **expression-list** ها می‌توانند شامل بيشتر از يك مقدار بوده و در اين صورت برای جدا کردن آن مقادير از يكديگر از علامت ويرگول استفاده می‌شود.
برای واضح‌تر شدن بحث به مثال زير توجه نمایيد:

```

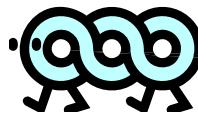
1 Select Case Val(Txt_Grade.Text)
2 Case 20, 19
3   MsgBox "Excellent :)"
4 Case 18, 17
5   MsgBox "Good :|"
6 Case Else
7   MsgBox "Bad :("
8 End Select

```

در ابتدا مقدار وارد شده در جعبه‌ی متن يعني **Val(Txt_Grade.Text)** در نظر گرفته می‌شود. در صورتی که اين مقدار برابر با عدد 20 و يا 19 باشد، دستور خط 3 يعني در واقع هرآن چيزی که برای اين دو حالت در نظر گرفته شده است، اجرا می‌شود. به همين ترتيب اگر مقدار جعبه‌ی متن برابر

18 و یا 17 باشد، دستور خط 5 اجرا می‌شود و درنهایت اگر مقدار جعبه‌ی متн هیچ‌کدام از مقادیر ذکر شده در بالا نباشد، دستورات مرتبط با Else یعنی دستور خط 7 اجرا می‌شود.

از مزیت‌های استفاده از ساختار Select ... Case ... If ... Then ... Elseif ... این است که مقداری که در جلوی عبارت Case قرار می‌گیرند باید همگی از یک نوع باشند. در نتیجه استفاده از این ساختار زمانی توصیه می‌شود که شرط‌های مساله برای حالات و مقادیر مختلف یک متغیر و یا خصوصیت مدنظر باشد.



۴.۳. ساختارهای حلقه

ساختارهای حلقه برای اجرا کردن بخشی از برنامه به دفعات، کاربرد دارند. در این بخش با دو ساختار حلقه مهم در ویژوال بیسیک آشنا خواهیم شد.

For ... Next .۱.۲.۳.۴

از این نوع حلقه زمانی استفاده می‌کنیم که تعداد دفعات تکرار از قبل مشخص باشد. در واقع این حلقه دارای یک شمارنده است که وظیفه آن شمارش تعداد دفعات تکرار است:

For مقدار نهایی شمارنده **To** مقدار اولیه شمارنده = نام شمارنده

statements

Next نام شمارنده

مقدار شمارنده در حلقه فوق هر بار ۱ واحد اضافه می‌شود. در صورتی که بخواهیم مقدار شمارنده هر بار K واحد اضافه شود، باید از عبارت Step K استفاده نمود.

For مقدار نهایی شمارنده **To** مقدار اولیه شمارنده = نام شمارنده **Step K**

statements

Next نام شمارنده

 نکته :

توجه کنید که مقدار K در عبارت Step K می‌تواند یک عدد منفی باشد. در این حالت شمارنده به صورت معکوس عمل می‌کند. برای روشن تر شده بحث به مثال زیر توجه نمایید:

For i = 10 To 2 Step -2

...

Next i

در این حلقه شمارنده ا به ترتیب مقادیر 10، 8، 6، 4، 2 را اختیار می‌کند.

برای مثال زمانی که می‌خواهیم مجموع اعداد از 1 تا 100 را محاسبه کنیم مطمئن هستیم که تعداد دفعات تکرار حلقه برابر 100 است. بنابراین می‌توانیم از این نوع حلقه استفاده نماییم. به مثال زیر دقت کنید:

 نمونه :

For i = 1 To 100

Bank = Bank + i

Next i

While ... Wend .۲.۳.۴

فرم کلی ساختار While به صورت زیر است:

While condition

statements

Wend

در این نوع حلقه تا زمانی که شرط condition برقرار باشد، تکرار دستورات داخل حلقه (statements) ادامه می‌یابد.

پیاده‌سازی حلقه‌ی For به کمک حلقه‌ی While

از آنجایی که شرط حلقه While می‌تواند هر عبارت منطقی باشد، در نتیجه هر حلقه For را می‌توان با کمک حلقه While پیاده‌سازی نمود، اما شاید برعکس آن در همه موارد به سادگی امکان پذیر نباشد.

```
For i = n To m Step k
...
Next i
```



```
i = n
While i <= m
...
i = i + k
Wend
```

۴.۴. تمرین

ساختمانی حلقه Do ... Loop: علاوه بر دو ساختار حلقه‌ای که در بالا دیده شد، ساختار حلقه دیگری با نام Do ... Loop در ویژوال بیسیک وجود دارد که می‌تواند به صورت‌های مختلف مورد استفاده قرار بگیرد. سعی کنید با استفاده از Help ویژوال بیسیک اطلاعات لازم را در رابطه با این ساختار بدست آورید.

درباره دستورات Exit For و Do ... Loop تحقیق کنید. برای این کار نیز می‌توانید از Help زبان ویژوال بیسیک استفاده نمایید.

گام پنجم

دوره‌ای برگرافیک

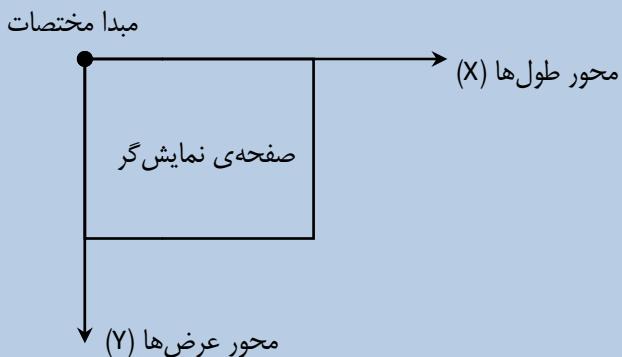


۱.۵ آشنایی

در این جلسه قصد داریم تا دوره‌ای بر گرافیک کامپیوتری و دستورات مربوط به آن داشته باشیم. همان‌طور که می‌دانید، صفحه نمایش از تعداد زیادی نقطه تشکیل شده است. تصاویر با تنظیم رنگ هر یک از این نقاط شکل می‌گیرند. در واقع، هنگامی که شما از دستوری استفاده می‌کنید که خطی با رنگ قرمز رسم کند، این دستور با تغییر رنگ نقاط واقع بر آن خط باعث ظاهر شدن خط مورد نظر بر روی صفحه نمایش گر می‌شود.

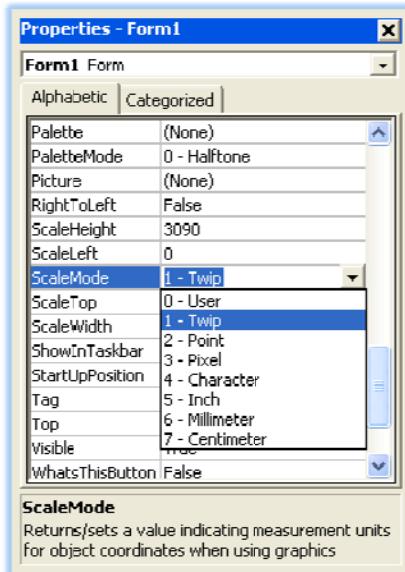


توجه: مبدأ مختصات در صفحه نمایش گر در گوشه بالا سمت چپ واقع شده است. شکل زیر نحوه قرارگیری مبدأ و محورهای مختصات را نشان می‌دهد.

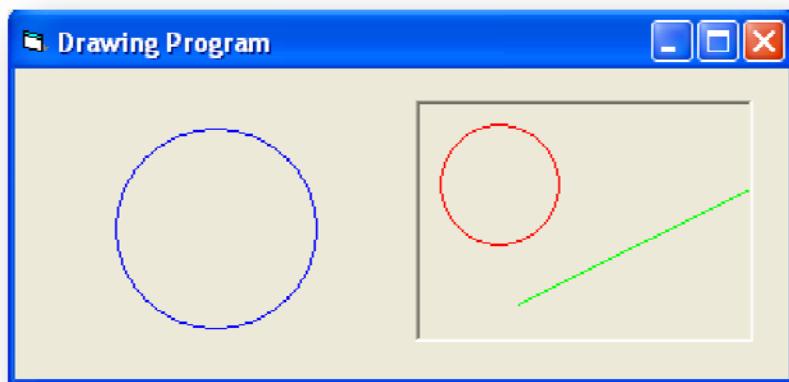


قبل از پرداختن به دستورات گرافیکی و نحوه استفاده از آن‌ها لازم است درباره مقیاس اندازه‌گیری مختصات به نکاتی اشاره کنیم. در زبان بیسیک مختصات نقاط صفحه بر اساس واحد پیکسل (Pixel) اندازه‌گیری می‌شوند. برای مثال زمانی که گفته می‌شد در مختصات (100,100) نقطه‌ای به رنگ آبی رسم شود بدین معنا بود که یک نقطه آبی در محلی که به اندازه 100 پیکسل از محور افقی و عمودی فاصله دارد، رسم شود. اما در زبان ویژوال بیسیک مقیاس مختصات می‌تواند علاوه بر پیکسل، سانتی‌متر، میلی‌متر، اینچ ... بوده و این مقیاس به کمک خصوصیت ScaleMode قابل انتخاب می‌باشد. برای مثال اگر خصوصیت ScaleMode را برابر Inch - 5 تنظیم کرده باشید و نقطه‌ای در مختصات (1,1) رسم کنید، این نقطه در فاصله‌ی 1 اینچی از محورهای افقی و عمودی قرار خواهد

گرفت. پیشنهاد می‌کنیم که مقدار خصوصیت ScaleMode را برابر 3 قرار دهید تا واحد اندازه‌گیری برابر پیکسل شود.



توجه به این نکته ضروری است که برای انجام کارهای گرافیکی (از جمله رسم اشکال، قرار دادن تصاویر و...) می‌توان از فرم و یا جعبه‌ی تصویر استفاده نمود. برای مثال شما می‌توانید مانند شکل زیر در داخل فرم و یا جعبه‌ی تصویر خط و یا دایره رسم نمایید.



معمولًاً برای کارهای گرافیکی از جعبه‌ی تصویر استفاده می‌شود. به همین جهت در ادامه به توضیحاتی درباره این کنترل خواهیم پرداخت.

۲.۵. جعبه‌ی تصویر (Picture Box) چیست؟

این کنترل برای نمایش تصاویر و انجام کارهای گرافیکی به کار می‌رود. در زیر به بررسی بعضی خصوصیات (Properties) و همچنین نکات قابل توجه در رابطه با جعبه‌ی تصویر می‌پردازیم. پیشنهاد می‌کنیم برای روشن‌تر شدن موضوع، پس از معرفی هریک از خصوصیات، آن‌ها را در محیط ویژوال بیسیک مورد آزمایش و بررسی قرار دهید.

۲.۵.۱. قرار دادن تصویر در جعبه‌ی تصویر (خصوصیت Picture)

برای این‌که تصویری را در جعبه‌ی تصویر قرار دهید کافی است از پنجره خصوصیات، مقدار خصوصیت Picture را انتخاب کرده و مسیر فایل تصویر مورد نظر را در آن وارد کنید.

۲.۵.۲. خصوصیت AutoSize

اگر مقدار این خصوصیت True باشد، جعبه‌ی تصویر به‌طور خودکار به اندازه تصویر در می‌آید ولی اگر مقدار این خصوصیت برابر False باشد چنین نخواهد شد. در شکل زیر دو عدد جعبه‌ی تصویر بر روی یک فرم وجود دارد که خصوصیت AutoSize جعبه تصویر سمت چپ برابر True و دیگری برابر False است. توجه کنید زمانی که مقدار خصوصیت AutoSize برابر False باشد، اندازه جعبه‌ی تصویر می‌تواند بزرگ‌تر و یا کوچک‌تر از تصویر باشد.



۳.۲.۵ خصوصیت ScaleMode

همان‌طور که توضیح داده شد، خصوصیت ScaleMode واحد اندازه‌گیری را تعیین می‌کند. با توجه به این که هم بر روی فرم و هم جعبه‌ی تصویر می‌توان اشکال گرافیکی رسم نمود، هر دو دارای خصوصیت ScaleMode هستند که بهتر است هر دو (فرم و جعبه‌ی تصویر) یک مقدار داشته باشند.

۴.۲.۵ خصوصیت‌های Height و Width

خصوصیت Width طول جعبه‌ی تصویر را بر حسب واحد اندازه‌گیری (ScaleMode) فرم ذخیره می‌کند. یعنی اگر واحد اندازه‌گیری فرم برابر میلی‌متر باشد، طول جعبه‌ی تصویر را بر حسب میلی‌متر می‌دهد. بنابراین اگر می‌خواهید که طول جعبه‌ی تصویر را بر حسب Pixel به دست بیاورید، باید خصوصیت ScaleMode فرم برابر 3 - Pixel باشد.

همچنین Height عرض جعبه‌ی تصویر را بر حسب واحد اندازه‌گیری فرم ذخیره می‌کند.

نکته:

اگر خصوصیت AutoSize مربوط به جعبه‌ی تصویر برابر True باشد آنگاه طول و عرض تصویر داخل جعبه‌ی تصویر برابر خصوصیت‌های Width و Height مربوط به جعبه‌ی تصویر می‌باشد. در نتیجه برای به دست آوردن طول و عرض تصویری که در داخل یک جعبه‌ی تصویر به نام Pic_MyPic وجود دارد، ابتدا خصوصیت AutoSize مربوط به Pic_MyPic را برابر True قرار داده و سپس از دو خط برنامه زیر استفاده کنید:

```
W = Pic_MyPic.Width
H = Pic_MyPic.Height
```

۵.۲.۵ خصوصیت AutoRedraw

این خصوصیت می‌تواند یکی از دو مقدار True و False را بپذیرد. هر جعبه‌ی تصویر دارای یک حافظه گرافیکی می‌باشد. هر چیزی که بر روی یک جعبه‌ی تصویر رسم می‌کنید، در حافظه گرافیکی آن قرار می‌گیرد. این حافظه تحت برخی شرایط پاک می‌شود. برای مثال اگر چیزی بر روی

یک جعبه‌ی تصویر رسم کرده باشید و سپس فرم برنامه‌تان را کوچک نموده و دوباره به حالت اول بازگردانید، تصویر رسم شده در جعبه‌ی تصویر را نمی‌بینید زیرا حافظه گرافیکی مربوط به جعبه‌ی تصویر پاک شده است. اگر می‌خواهید که حافظه گرافیکی جعبه‌ی تصویر پاک نشود باید مقدار خصوصیت AutoRedraw را برابر True قرار دهید.

تمرین:

یک جعبه‌ی تصویر بر روی فرم گذاشته و از طریق خصوصیت Picture یک تصویر در آن قرار دهید. مطمئن باشید که مقدار خصوصیت AutoRedraw برابر False است. سپس برنامه را با کمک کلید F5 اجرا نموده و پس از مشاهده تصویر مورد نظر در جعبه‌ی تصویر، فرم برنامه‌تان را به حالت کوچک شده درآورده و بار دیگر به حالت اول بازگردانید. چه خواهید دید؟ اکنون برنامه را متوقف کرده و مقدار خصوصیت AutoRedraw را برابر True قرار دهید. آزمایش را دوباره تکرار کرده و نتیجه را مشاهده کنید.

نکته:

اگر برنامه‌ای برای رسم و تغییر عکس جعبه‌ی تصویر را در Form_Load می‌نویسید، حاصل اجرای آن را بر روی جعبه‌ی تصویر نمی‌بینید، برای مشاهده آن باید خصوصیت AutoRedraw را برابر True قرار دهید.

Pset .۵.۳ دستور

PSET شماره رنگ ، (عرض نقطه ، طول نقطه)

دستور Pset(x,y),C یک نقطه در مختصات (x,y) و با رنگ شماره C، بر روی فرم رسم می‌کند. برای این که بتوانیم بر روی یک جعبه‌ی تصویر نقطه‌ای رسم کنیم باید ابتدا نام جعبه‌ی تصویر را نوشته،

سپس یک علامت نقطه و در نهایت از دستور `Pset` استفاده نماییم، زیرا همان‌طور که قبلاً نیز توضیح داده شد، برای مشخص کردن این که یک خصوصیت و یا متند مربوط به چه کنترلی می‌باشد، از علامت نقطه استفاده می‌کنیم. برای مثال اگر نام جعبه‌ی تصویر برابر `Pic_MyPic` باشد، باید بنویسیم:

`Pic_MyPic.PSet (100, 100), 65535`

تمرين: یک جعبه‌ی تصویر و یک دکمه فرمان بر روی فرم قرار دهید. برنامه‌ای بنویسید که با کلیک کردن بر روی دکمه فرمان، بدون استفاده از دستور `Line`، یک خط از مختصات (10,10) تا مختصات (110,10) در جعبه‌ی تصویر رسم نماید.
راهنمایی: یک خط از تعدادی نقاط به هم پیوسته تشکیل شده است.

تمرين: یک جعبه‌ی تصویر و یک دکمه فرمان بر روی فرم قرار دهید. برنامه‌ای بنویسید که با کلیک کردن بر روی دکمه فرمان، بدون استفاده از دستور `Line`، یک مستطیل توپر از مختصات (10,10) تا مختصات (110,60) در جعبه‌ی تصویر رسم نماید.

(10,10)



{110,60}

5.4. دستور Line

شماره رنگ ، (عرض نقطه انتهای ، طول نقطه انتهای) – (عرض نقطه ابتدای ، طول نقطه ابتدای) LINE

دستور `C` Line $(x_1,y_1)-(x_2,y_2)$ خطی را بین دو نقطه (x_1,y_1) و (x_2,y_2) با رنگ `C`، بر روی فرم رسم می‌نماید. مانند قبل برای این که بتوانیم در یک جعبه‌ی تصویر خطی را رسم کنیم، باید نام جعبه‌ی تصویر را به همراه علامت نقطه قبل از دستور `line` قرار دهیم. برای مثال اگر نام جعبه‌ی تصویر برابر `Pic_MyPic` باشد، باید بنویسیم:

```
Pic_MyPic.Line (10, 10)-(110, 10), 255
```

تمرین: یک جعبه‌ی تصویر و یک دکمه فرمان بر روی فرم قرار دهید. برنامه‌ای بنویسید که با کلیک کردن بر روی دکمه فرمان، با استفاده از دستور `Line`، یک مستطیل توپر از مختصات $(10,10)$ تا مختصات $(110,60)$ در جعبه‌ی تصویر رسم نماید.

5.5. دستور Circle

CIRCLE شماره رنگ ، شعاع ، (عرض مرکز ، طول مرکز)

دستور `C` Circle(x,y,r,C) دایره‌ای به به مرکز (x,y) و به شعاع r و با رنگ `C` رسم می‌نماید. برای این که بتوانیم دایره را در یک جعبه‌ی تصویر رسم نماییم، کافی است نام جعبه‌ی تصویر را به همراه علامت نقطه، قبل از دستور `Circle` قرار دهیم. در غیر این صورت مطابق قبل دایره را بر روی فرم رسم نماید. مثال:

```
Pic_MyPic.Circle (100, 100), 50, 255
```

تمرین: یک جعبه‌ی تصویر و یک دکمه‌ی فرمان بر روی فرم قرار دهید. برنامه‌ای بنویسید که با کلیک کردن بر روی دکمه‌ی فرمان، یک دایره توپر به مرکز $(100,100)$ و شعاع 50 در جعبه‌ی تصویر رسم نماید.

۵.۶. دستور Point

(عرض نقطه ، طول نقطه) **POINT** = متغیر برای ذخیره کردن رنگ نقطه

دستور (x,y) Point یکی از پرکاربردترین دستورات گرافیکی مخصوصاً در تمرین‌های بعدی این کتاب است. به کمک این دستور می‌توان شماره رنگ نقطه‌ای در مختصات (y,x) را بدست آورد. در واقع این دستور یک عدد به عنوان خروجی بازمی‌گرداند که شماره رنگ مختصات مورد نظر است.

تمرین: یک جعبه تصویر بر روی فرم قرار داده و به کمک خصوصیت Picture یک تصویر در داخل آن جای دهید. سپس به کمک دستور Point مقدار رنگ تعدادی از نقاط آن را بدست آورید. برای مثال کافی است که دو خط دستور زیر را برای رخداد کلیک یک دکمه فرمان بنویسید:

```
C = Point(150, 150)
MsgBox C
```

پس از کلیک کردن بر روی دکمه فرمان مورد بحث، شماره رنگ نقطه (150,150) در یک جعبه پیغام ظاهر می‌شود.

۷.۵. درباره رنگ‌ها بیشتر بدانیم

در ویژوال بیسیک تعداد رنگ‌های موجود وابسته به تعداد رنگ‌های سیستم عامل ویندوز می‌باشد. اگر کامپیوتری که در اختیار دارید از حالت 32Bit و یا 24bit پشتیبانی نماید، در این صورت به $16,777,215 = 2^{24}$ رنگ دسترسی خواهید داشت.

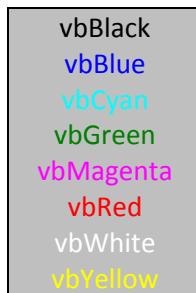
در ویژوال بیسیک به چندین طریق می‌توانیم از این رنگ‌ها استفاده کنیم که در زیر برخی از آن‌ها را مطرح خواهیم کرد:

- شماره هر رنگ را به خاطر بسپارید، همان‌طور که در بیسیک شماره رنگ‌ها را حفظ کرده بودید. فقط در این‌جا تعداد رنگ‌ها خیلی خیلی بیشتر است و چنین کاری تقریباً امکان‌پذیر نمی‌باشد.

برای مثال دستور زیر یک دایره با رنگ 16077700 رسم می‌کند:

`Circle(100,100),50, 16077700`

- ویژوال بیسیک برای برخی از رنگ‌های پر استفاده از ثابت‌ها استفاده می‌کند. به نمونه ثابت‌های زیر توجه نمایید:



در واقع ثابت `vbYellow` برابر با عدد 65535 تعریف شده است که شماره رنگ زرد می‌باشد.

بنابراین هر دو دستور زیر در واقع معادل بوده و منجر به رسم دایره زرد رنگ می‌شوند:

`Circle(100,100),50, 65535`

`Circle(100,100),50, vbYellow`

همان‌طور که می‌دانیم هر رنگ که در صفحه به نمایش گذاشته می‌شود ترکیبی از سه نور اصلی قرمز، سبز و آبی می‌باشد. بنابراین مخلوط کردن رنگ (نور) های اصلی یکی از بهترین راه‌های تولید رنگ است. تابع RGB این کار را انجام می‌دهد. این تابع مقدار (شدت) هر کدام از سه رنگ اصلی قرمز، سبز و آبی را گرفته و سپس شماره رنگی که حاصل ترکیب این سه رنگ اصلی است را برمی‌گرداند. هر کدام از مقادیر رنگ‌های قرمز، آبی و سبز می‌توانند بین صفر و ۲۵۵ باشند. یعنی در مجموع $2^8 \times 2^8 \times 2^8 = 2^{24}$ تعداد رنگ خواهیم داشت.

بنابراین سه دستور زیر معادل بوده و دایره‌ای به رنگ زرد رسم می‌کنند:

`Circle(100,100),50, 65535`

`Circle(100,100),50, vbYellow`

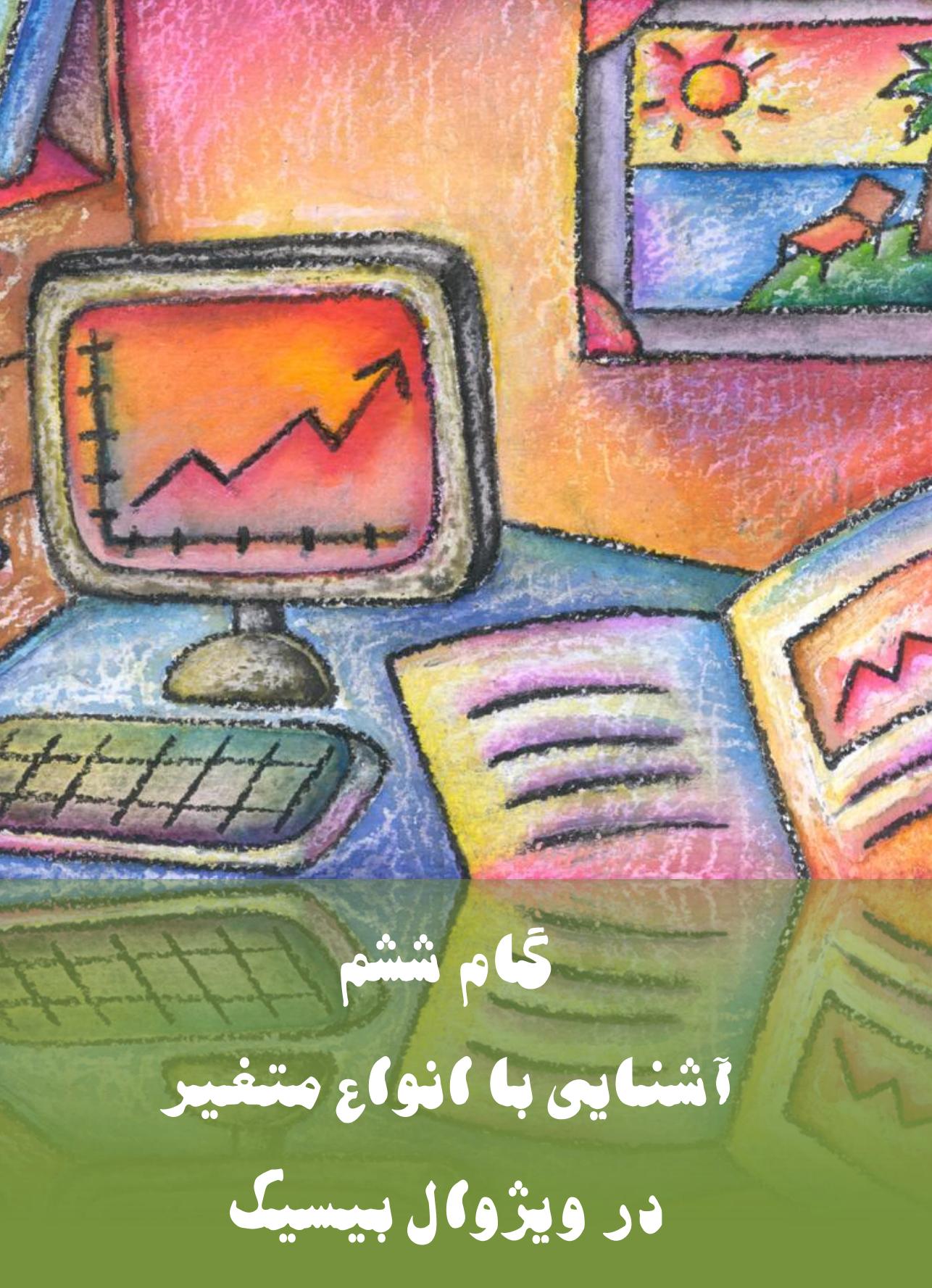
`Circle (100, 100), 50, RGB(255, 255, 0)`

توجه: توجه داشته باشید که تابع RGB یک عدد برمی‌گرداند. برای آزمایش این موضوع از



دستور زیر کمک بگیرید:

`MsgBox RGB(50,100,200)`



گام ششم

آشنایی با انواع متغیر

در ویژوال بیسیک

۶. ۱. آشنایی

اگر یادتان باشد در زبان بیسیک دو نوع متغیر عددی و کلمه‌ای داشتیم. در مثال زیر Name\$ یک متغیر کلمه‌ای و Age یک متغیر عددی است.

Input Name\$

Input Age

Print Name\$ + " is " + Str\$(Age) + " years old."

۶. ۲. انواع داده‌ای

در دنیای ویژوال بیسیک انواع بیشتری متغیر نسبت به زبان بیسیک وجود دارد، که در زیر فهرستی از آن‌ها را مشاهده می‌کنید.

توضیحات	حافظه‌ی مصرفی	نوع متغیر
False یا True	۲ بایت	Boolean
اعداد صحیح از -32786 تا 32786	۲ بایت	Integer
اعداد صحیح در حدود ±2,000,000,000	۴ بایت	Long
اعداد اعشاری با دقت کم	۴ بایت	Single
اعداد اعشاری با دقت بالا	۸ بایت	Double
متن	یک بایت برای هر کاراکتر	String

برای تعریف متغیرها می‌توان به صورت زیر عمل کرد:

Dim Name As String

Dim Age As Integer

Dim Weight As Single

همان‌طور که می‌دانیم از متغیرها برای نگهداری داده‌ها و اطلاعات در برنامه استفاده می‌شود. با توجه به نوع داده‌ای که قرار است در یک متغیر ذخیره شود، نوع آن را انتخاب می‌نماییم. برای مثال اگر

طمئن هستیم که داده‌ی مورد نظر از نوع عدد صحیح و کوچکتر از 1000 می‌باشد، متغیر مرتبط با آن را از نوع Integer در نظر می‌گیریم.

۶. حوزه‌ی متغیرها

همان‌طور که در جلسات قبل دیده‌اید، برنامه‌های شما از تعدادی زیربرنامه (مانند زیر) تشکیل شده‌اند.

```
Private Sub Cmd_Exit_Click()
    End
End Sub
```

در صورتی که یک متغیر در یک زیر برنامه تعریف شود، تنها در آن زیر برنامه شناخته می‌شود. برای درک بهتر این موضوع، برنامه‌ی زیر را ایجاد نمایید.

دو دکمه‌ی فرمان را بر روی صفحه قرار داده و خصوصیات آن‌ها را طبق دو جدول زیر تنظیم کنید.

نام	مقدار
Name	Cmd_Count
Caption	Count

نام	مقدار
Name	Cmd_Msg
Caption	Msg

برای رخداد Click دکمه فرمان Cmd_Count، قطعه برنامه‌ی زیر را بنویسید.

```
Private Sub Cmd_Count_Click()
    Dim K As Integer
    K = K + 1
End Sub
```

نکته

کافی است بر روی دکمه‌ی فرمان دو کلیک نمایید تا پنجره‌ی مربوط به برنامه‌نویسي باز شده و مطمئن شوید که برای رخداد کلیک، برنامه‌نویسي می‌کنید. در صورتی که فراموش کرده‌اید که چگونه می‌توان این کار را انجام داد، به شما پیشنهاد می‌کنیم به درس‌های جلسه‌ی قبل نگاهی بیاندازید.)

برای رخداد دکمه‌ی فرمان Click، قطعه برنامه‌ی زیر را بنویسید.

```
Private Sub Cmd_Msg_Click()
    MsgBox K
End Sub
```

توجه

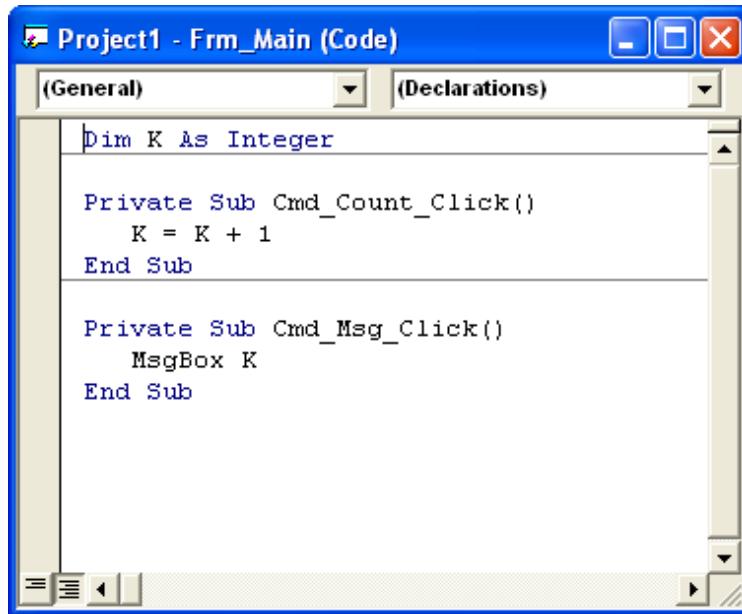
برنامه را با کمک کلید F5 اجرا کنید. چندین بار بر روی دکمه‌ی فرمان Count کلیک نمایید. اکنون بر روی دکمه‌ی فرمان Msg کلیک نمایید. همان‌طور که مشاهده می‌کنید، عدد 0 در جعبه‌ی پیغام ظاهر می‌شود. بار دیگر بر روی دکمه‌ی فرمان Count و سپس Msg کلیک نمایید. با توجه به این که در زیر برنامه‌ی مربوط به دکمه‌ی Count، مقدار K (به ازای هر بار کلیک شدن) 1 واحد افزایش می‌یابد، اما چرا همواره عدد 0 در جعبه‌ی پیغام ظاهر می‌شود؟

اکنون برنامه‌ها را به صورت زیر تغییر دهید.

```
Dim K As Integer
```

```
Private Sub Cmd_Count_Click()
    K = K + 1
End Sub
```

```
Private Sub Cmd_Msg_Click()
    MsgBox K
End Sub
```



The screenshot shows the Microsoft Visual Basic Editor window titled "Project1 - Frm_Main (Code)". The "General" tab is selected. The code in the editor is:

```

Dim K As Integer

Private Sub Cmd_Count_Click()
    K = K + 1
End Sub

Private Sub Cmd_Msg_Click()
    MsgBox K
End Sub

```

دقت کنید زمانی که نشان‌گر چشمکزن، بر روی خط Dim K As Integer واقع شده است، مقادیر دو جعبه‌ی بالای پنجره‌ی برنامه‌نویسی برابر (General) و (Declarations) خواهد بود. به شکل قبل نگاه کنید.

بار دیگر برنامه را با کمک کلید F5 اجرا کنید. یک بار بر روی دکمه‌ی فرمان Count کلیک نمایید. سپس بر روی دکمه‌ی فرمان Msg کلیک نمایید. همان‌طور که مشاهده می‌کنید، عدد 1 در جعبه‌ی پیغام ظاهر می‌شود. بار دیگر بر روی دکمه‌ی فرمان Count و سپس Msg کلیک نمایید. این بار عدد 2 در جعبه‌ی پیغام ظاهر می‌شود. چرا؟

اگر به توضیحات ابتدای این بخش دقت کرده باشید، گفتیم که در صورتی که یک متغیر در یک زیر برنامه تعریف شود، تنها در آن زیر برنامه شناخته می‌شود.

در واقع هر زیر برنامه دارای متغیرهای محلی (Local) برای خودش می‌باشد. یعنی در حالت اول دو متغیر K که در زیر برنامه استفاده شده‌اند، دو متغیر جداگانه و مستقل از هم محسوب می‌شوند.

```
Private Sub Cmd_Count_Click()
```

```
    Dim K As Integer
```

```
    K = K + 1
```

```
End Sub
```

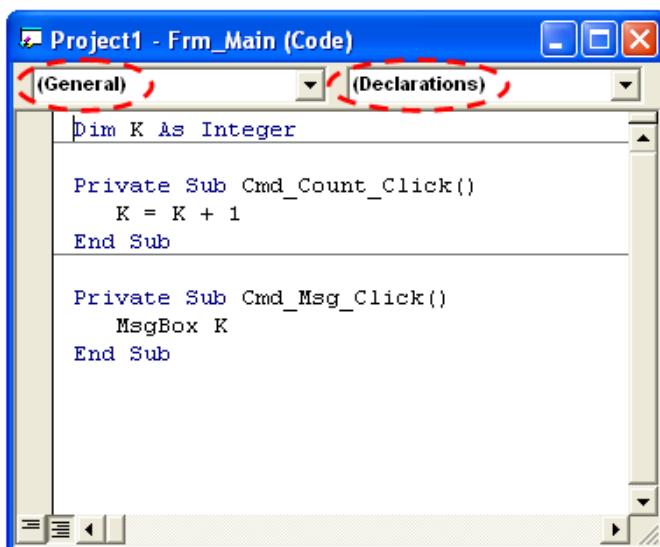
```
Private Sub Cmd_Msg_Click()
```

```
    MsgBox K
```

```
End Sub
```

اگر چه نام دو متغیر در دو زیر برنامه‌ی رو به رو یکسان است، اما در واقع این دو از یکدیگر متمایز بوده و دو فضای متفاوت از حافظه به آن‌ها اختصاص داده شده است.

در صورتی که قصد داشته باشید که هر دو متغیر تعریف شده در دو زیر برنامه، دارای یک فضای حافظه باشند و به عبارت ساده‌تر یکی باشند، باید تعریف متغیر مورد نظر را در بالای صفحه‌ی مربوط به برنامه‌نویسی قرار دهید. مطابق شکل زیر:



به مقادیر دو جعبه بالای پنجره برنامه‌نویسی، (General) و (Declarations) دقت کنید.

با قرار دادن تعریف متغیر در قسمت General Declarations، حوزه‌ی تعریف متغیر به صورت سراسری (Global) خواهد بود و مقدار خود را تا پایان اجرای برنامه حفظ می‌کند. در نتیجه زمانی که

این برنامه اجرا می‌شود، شاهد اعدادی بزرگتر از صفر (بسته به تعداد بار کلیک شدن دکمه‌ی فرمان) در جعبه‌ی پیغام (Message Box) خواهیم (Count) بود.

توجه

با توجه به این‌که این‌که تعریف صریح متغیر با عبارت Dim در ویژوال بیسیک اجباری نیست، در صورتی که متغیری بدون تعریف صریح، در یک زیر برنامه استفاده شود، یک متغیر محلی برای آن زیر برنامه محسوب خواهد شد.

نکته

دو نکته‌ی مهم در رابطه با تعریف متغیرها در برنامه‌نویسی حرفه‌ای وجود دارد که لازم است به آن‌ها توجه داشته باشیم.

- نام متغیرها باید معنادار باشد: برای مثال اگر از متغیری برای ذخیره کردن مقدار حاصل جمع استفاده می‌کنید، می‌توانید نام Sum را برای آن در نظر بگیرید. ممکن است در برنامه‌های کوچکی که تاکنون نوشته‌اید، ضرورت این موضوع را احساس نکرده باشید؛ اما با بزرگتر شدن اندازه‌ی برنامه‌ها و در نتیجه افزایش تعداد متغیرها، استفاده از نام معنادار برای متغیرها اجتناب ناپذیر است. اگر این موضوع را رعایت نکنید، در هنگام رفع ایرادهای برنامه‌هایتان، دچار مشکل خواهد شد.
- گفتیم که ویژوال بیسیک اجباری برای تعریف صریح متغیر با کمک عبارت Dim ندارد، مگر اینکه در قسمت مربوط به تعاریف (General Declarations)، عبارت Option Explicit را قرار دهید. با قرار گرفتن این عبارت در محل مربوطه، ویژوال بیسیک شما را مجبور خواهد کرد تا متغیرها را قبل از استفاده، به صورت صریح تعریف نمایید. شاید در ابتدا فکر کنید که چنین کاری جز این که بر سختی برنامه‌نویسی بیفزاید، فایده‌ی دیگری ندارد. اگر چنین فکری می‌کنید، مطمئن باشید که در اشتباه هستید. برای روشن شدن این موضوع، به مثال زیر توجه نمایید:

```
For i = 1 To n
    Bank = Bnk + i
Next i
```

در این قطعه برنامه، در یک جا به طور اشتباه به جای Bank نوشته شده است Bnk. در حالت معمول، چون ویژوال بیسیک شما را مجبور به تعریف متغیرها نمی‌کند، برنامه هیچ خطایی را اعلام نمی‌کند. زیرا فرض می‌کند که شما قصد استفاده از دو متغیر جداگانه به نام‌های Bank و Bnk را داشته‌اید. اما در صورت استفاده از عبارت Option Explicit و در نتیجه تعریف صریح متغیر Bank و به دلیل نبودن تعریفی برای متغیر Bnk، برنامه پیغام خطایی را عدم تعریف یک متغیر اعلام می‌کند، که شما به راحتی می‌توانید ایراد مربوطه را برطرف سازید. در صورتی که چنین پیغام خطایی ظاهر نمی‌شد، مجبور به جستجو برای پیدا کردن ایراد کار بودید؛ البته بدون آن که هیچ‌گونه سر نخی از مشکل داشته باشید.

۶.۴. دوره‌ی حیات (طول عمر) متغیرها

در بخش قبل با حوزه‌ی متغیرها آشنا شدید. نکته‌ی مهم دیگر در رابطه با متغیرها، دوره‌ی حیات و یا طول عمر آن‌هاست. دوره‌ی حیات، مدت زمانی است که یک متغیر، مقدار خود را حفظ می‌کند.

- متغیرهای سراسری یا Global، از ابتدای اجرای برنامه تا پایان اجرای آن، مقدار خود را حفظ کرده و در نتیجه دوره‌ی حیات آن‌ها برابر است با دوره‌ی حیات برنامه.
- متغیرهای محلی یا Local، که با عبارت Dim تعریف می‌شوند، بعد از پایان اجرای زیر برنامه از بین رفته و مقدار خود را از دست می‌دهند.

برنامه‌ی مربوط به دکمه‌ی فرمان Count را به صورت زیر بنویسید. برنامه را با دکمه F5 [F5] اجرا کنید و بر روی دکمه‌ی فرمان Count کلیک نمایید. خواهید دید هر تعداد باری که بر روی این دکمه کلیک کنید، تنها عدد صفر در جعبه‌ی پیغام ظاهر می‌شود. با توجه به توضیحات بالا، آیا می‌توانید دلیل آن را بیان کنید؟

```
Private Sub Cmd_Count_Click()
    Dim K As Integer
    MsgBox K
    K = K + 1
End Sub
```

با هر بار کلیک کردن بر روی دکمه‌ی فرمان Count، یک متغیر K با مقدار اولیه‌ی صفر تعریف شده و مقدار آن در جعبه‌ی پیغام (با کمک دستور MessageBox) نمایش داده می‌شود. سپس مقدار K یک واحد افزایش یافته و مقدار آن برابر ۱ می‌شود و در نهایت اجرای زیر برنامه به پایان می‌رسد. با به پایان رسیدن اجرای زیر برنامه، متغیر K از بین رفته و مقدار آخر خود (یعنی ۱) را از دست می‌دهد. بار بعدی که بر روی این دکمه‌ی فرمان کلیک می‌کنید، دوباره متغیر K با مقدار اولیه صفر تعریف شده و

حال برای این‌که یک متغیر محلی داشته باشیم که پس از اجرای زیر برنامه مقدار خود را از دست ندهد، چه باید بکنیم؟ سوال خود را به گونه‌ی دیگری مطرح می‌کنیم: برای داشتن یک متغیر با حوزه‌ی محلی، اما دوره‌ی حیات عمومی، چه راه حلی وجود دارد؟

راه حل این است که به‌جای تعریف متغیر محلی، با استفاده از عبارت Dim، متغیر محلی را با استفاده از عبارت Static (به معنای ایستا) تعریف نماییم. در این صورت اگر چه حوزه‌ی متغیر، همان زیر برنامه‌ای است که در آن تعریف شده است، اما با پایان یافتن اجرای زیر برنامه، مقدار خود را از دست نمی‌دهد.

نمونه

توجه به این نکته ضروری است که برای تعریف متغیرهای سراسری، نمی‌توان از عبارت Static استفاده نمود و البته چنین کاری اصلاً فاقد معنی است. زیرا دوره‌ی حیات متغیر سراسری برابر با دوره‌ی حیات کل برنامه است.



تمرین

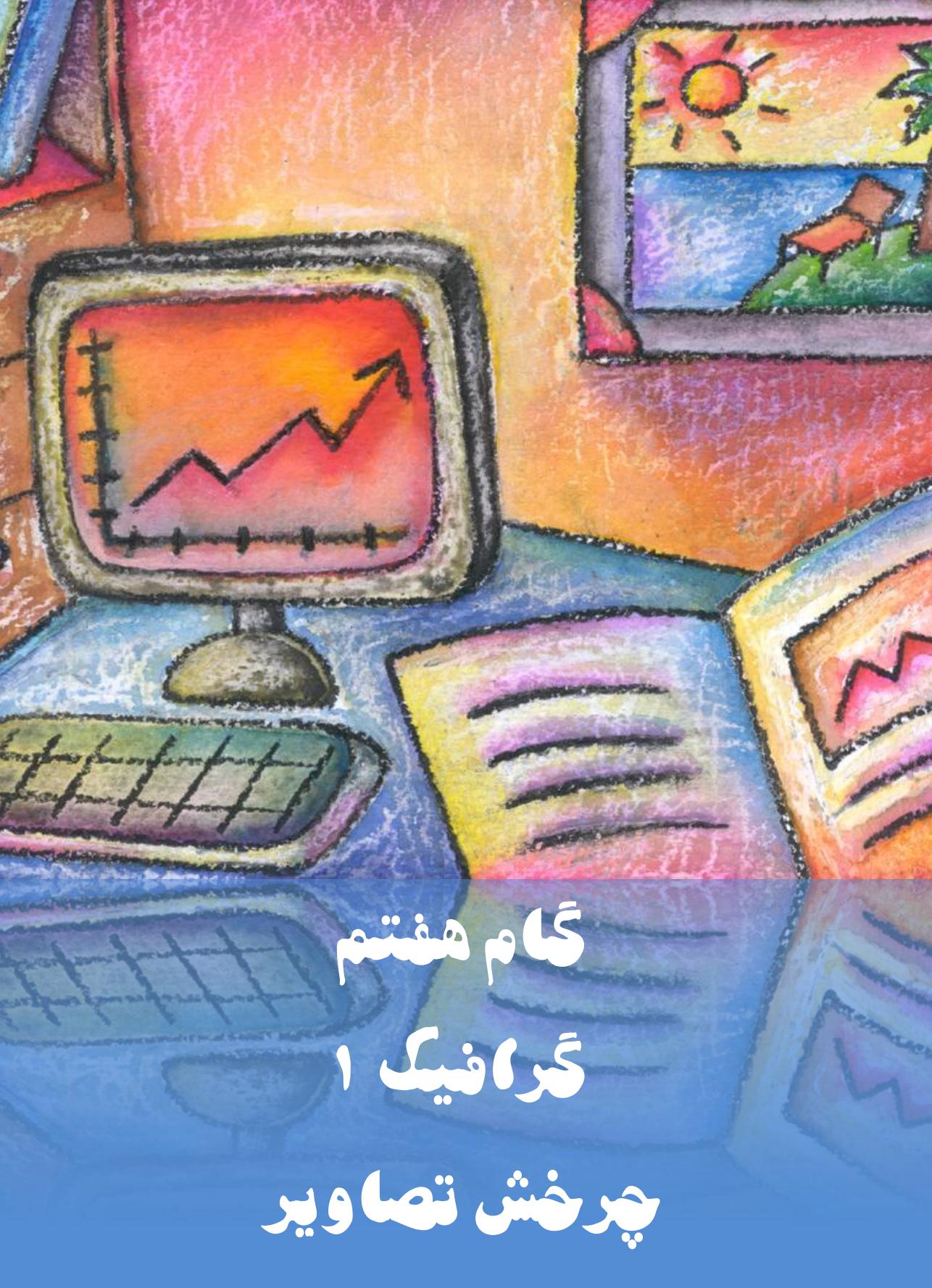
برنامه‌ی قبل را به صورت روبرو، اصلاح نمایید و نتیجه را بررسی کنید.

```
Private Sub Cmd_Count_Click()
    Static K As Integer
    MsgBox K
    K = K + 1
End Sub
```

تمرین

فرض کنید متغیری را به صورت سراسری تعریف کده و همچنین با نامی مشابه این متغیر، متغیری محلی را در یک زیر برنامه تعریف کرده‌ایم. به نظر شما در زیر برنامه‌ی مذکور، اولویت با کدامیک از متغیرهایست؟ برنامه‌ای بنویسید تا بتوانید این موضوع را آزمایش کده و نتیجه را گزارش نمایید.





گام هفتم

گرافیک ۱

چرخش تصاویر

۱.۷ آشنایی

در جلسه امروز می‌خواهیم چند پروژه گرافیکی کوچک اما جالب بنویسیم. بعد از آن که به محیط برنامه‌سازی ویژوال بیسیک وارد شدید، برای انجام پروژه‌های زیر دست به کار شوید.

۲.۷ پروژه ۱

ابتدا دو عدد جعبه‌ی تصویر (PictureBox) و یک عدد دکمه‌ی فرمان (CommandButton) روی فرم قرار دهید. در جعبه‌ی تصویر اول تصویری را بارگذاری (Load) نمایید. سپس برنامه‌ای را ایجاد کنید که با کلیک کردن بر روی دکمه‌ی فرمان، تصویر موجود در جعبه‌ی تصویر اول را در جعبه‌ی تصویر دوم رسم کند.

راهنمایی: برنامه شما باید رنگ هر نقطه‌ای از تصویر موجود در جعبه‌ی تصویر اول را بخواند و به ازای آن در جعبه‌ی تصویر دوم نقطه‌ای با همان رنگ در محل مناسب رسم کند.



۳.۷. پروژه ۲

ابتدا تصاویر زیر را به دقت بینید.

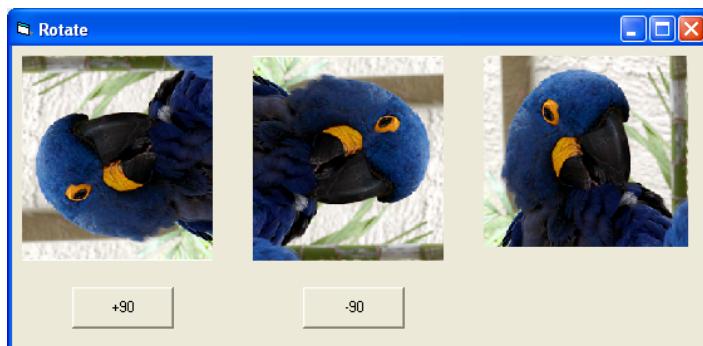


تصویر اصلی ۹۰ درجه در جهت
پادساعت‌گرد دوران داده است.

تصویر اصلی ۹۰ درجه در جهت
 ساعت‌گرد دوران داده شده است.

تصویر اصلی

ابتدا سه عدد جعبه‌ی تصویر بر روی فرم قرار داده و نامهای آن‌ها را بگذارید. تصویری را در Pic_1 قرار دهید. دو عدد دکمه‌ی فرمان نیز بر روی فرم قرار دهید و نامهای آن‌ها را Cmd_CCW و Cmd_CW قرار دهید. اکنون برنامه‌ای بنویسید که وقتی روی دکمه Cmd_CW کلیک کردیم، تصویر موجود در Pic_1 را در Pic_2 رسم کند بهصورتی که ۹۰ درجه در جهت ساعت‌گرد دوران داده شده باشد. بهطور مشابه، وقتی بر روی Cmd_CCW کلیک کردیم، تصویر موجود در Pic_1 را در Pic_3 رسم کند بهصورتی که ۹۰ درجه در جهت پادساعت‌گرد دوران داده شده باشد. پس از اجرا فرم برنامه‌تان باید مانند تصویر باشد:



تمرین:

برنامه‌ای بنویسید که یک تصویر را نسبت به محور افقی و عمودی معکوس نماید:
توجه کنید معکوس نسبت به محور افقی با دوران 180 درجه‌ای متفاوت است. برای واضح‌تر شدن
تمرین به شکل‌های زیر دقت کنید:



معکوس تصویر اصلی نسبت به محور
عمودی
افقی

تصویر اصلی



گام هشتم

کار کردن با ماوس

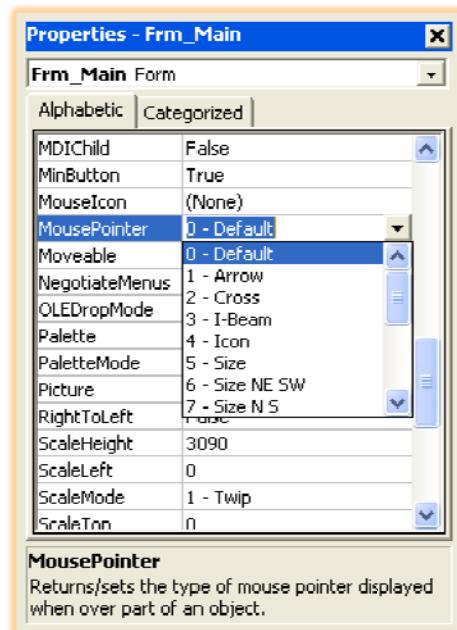


۱.۸ آشنایی

ماوس و صفحه کلید ابزارهایی برای ارتباط با برنامه‌های ویژوال بیسیک و یا هر رابط کاربری محسوب می‌شوند. به بیان دقیق‌تر ماوس و صفحه کلید دستگاه‌های ورودی یک سیستم کامپیوتری محسوب می‌شوند. از ماوس می‌توان برای کلیک کردن، دوکلیک کردن، جابه‌جا کردن و... استفاده نمود. در واقع حرکت دادن و یا فشردن و رها کردن دکمه‌های ماوس باعث رخدادهایی می‌شود که برنامه‌نویس می‌تواند برای آن‌ها برنامه ایجاد نموده تا در زمان وقوع رخداد مورد نظر، برنامه‌ی متناظرش اجرا شود. همچنین فشردن و رهاشدن دکمه‌های صفحه کلید به‌طور مشابه منجر به رخدادهایی می‌شود که می‌توان برای آن‌ها برنامه‌نویسی نمود. در این جلسه قصد داریم تا نحوه کار با ماوس و در یکی از جلسه‌های آینده نحوه کار با صفحه کلید در برنامه‌های ویژوال بیسیک را مورد بررسی قرار خواهیم داد.

۲.۸ خصوصیت Mouse Pointer برای تغییر ظاهر اشاره‌گر ماوس

فرم و اکثر کنترل‌هایی که با آن‌ها آشنا شده و یا خواهید شد، دارای خصوصیتی به نام MousePointer هستند. این خصوصیت تعیین کننده ظاهر اشاره‌گر ماوس بر روی فرم و یا کنترل مورد نظر است.



همان طور که در شکل صفحه‌ی قبل قابل مشاهده است، این خصوصیت می‌تواند مقادیر متفاوتی را به خود بگیرد. در این صورت ظاهر اشاره‌گر ماوس بر اساس مقدار انتخاب شده، بر روی فرم یا کنترل مورد نظر تغییر خواهد کرد.

یک کنترل دکمه‌ی فرمان (CommandButton) بر روی فرم برنامه قرار دهید. سپس مقدار خصوصیت MousePointer مربوط به این دکمه را برابر Size - 5 قرار دهید. اکنون برنامه‌تان را با کمک F5 اجرا نموده و اشاره‌گر ماوس را بر روی دکمه فرمان حرکت دهید.

تمرین:

بررسی کنید که ظاهر اشاره‌گر ماوس در برنامه‌ی قبل چه تغییری می‌کند؟

تمرین:

چگونه می‌توان تصویری دلخواه (البته با پسوندهای مجاز) را به عنوان اشاره‌گر ماوس برگزید؟ راهنمایی: خصوصیت‌های MousePointer و Mouselcon را بررسی نمایید.

۳.۸ انواع رخدادها

در این جدول تعدادی از رخدادهای پرکاربرد ماوس را مشاهده می‌کنید.

از آنجا که بارها برای این رخداد برنامه نوشته‌اید، برایتان بدیهی است این رخداد زمانی اتفاق می‌افتد که بر روی فرم و یا کنترل یکبار کلیک کرده باشیم.

Click

این رخداد زمانی اتفاق می‌افتد که عمل دوکلیک (دو تا کلیک پشت سرهم) بر روی فرم و یا کنترلی واقع شده باشد.

Double Click

هر زمان ماوس را بر فرم و یا کنترلی حرکت دهیم، رخداد MouseMove مربوط به آن فرم و یا کنترل اتفاق می‌افتد.

Mouse Move

این رخداد زمانی اتفاق می‌افتد که دکمه ماوس را بر روی فرم و یا کنترل فشار داده باشیم.

Mouse Down

این رخداد زمانی اتفاق می‌افتد که دکمه ماوس را بر روی فرم و یا کنترل رها کرده باشیم.

Mouse Up


نمونه

فرض کنید برای رخداد دکمه فرمان Cmd_MouseMove برنامه زیر را نوشتیم:

```
Private Sub Cmd_MouseMove(Button As Integer, Shift As Integer, X  
As Single, Y As Single)
```

```
    Frm_Main.BackColor = vbRed  
End Sub
```

در صورت حرکت ماوس بر روی دکمه فرمان با نام Cmd_MouseButton، رنگ زمینه فرم (با نام Frm_Main) قرمز می‌شود.

۴.۸. نکاتی درباره رخدادهای Mouse Up ، Mouse Down و Mouse Move

وقتی که می‌خواهید برای رخدادهای Mouse Up و Mouse Down و Mouse Move برنامه‌نویسی کنید، به عباراتی که ویژوال بیسیک به خودکار برای شما تولید می‌کند دقت کنید:

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y  
As Single)
```

```
End Sub
```

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y  
As Single)
```

```
End Sub
```

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As  
Single)
```

```
End Sub
```

تفاوتی بین آنچه در بالا آمده است با آنچه برای رخداد Click تولید می‌شود وجود دارد.

```
Private Sub Form_Click()
```

```
End Sub
```

بله، درست متوجه شده‌اید. در تعریف زیربرنامه‌های مربوط به رخدادهای Mouse Move تعدادی پارامتر در داخل پرانتز وجود دارد.

- **پارامترهای X و Y:** این دو پارامتر مختصات نقطه‌ای که رخداد واقع شده است را دارد. این دو مقدار در زیربرنامه مورد نظر قابل استفاده هستند.
- **پارامتر Button:** مقداری که این پارامتر در خود ذخیره می‌کند بیانگر این است که در هنگام وقوع رخداد کدامیک از دکمه‌های ماوس فشرده شده است. در واقع وضعیت دکمه‌های ماوس به صورت یک عدد Integer در متغیر Button ذخیره می‌شوند.

ثابت	مقدار	شرح
vbLeftButton	1	دکمه سمت چپ ماوس
vbRightButton	2	دکمه سمت راست ماوس
vbMiddleButton	4	دکمه وسط ماوس

برای مثال در صورتی که هنگام وقوع این رخداد مقدار پارامتر Button برابر عدد 1 باشد، بدین معنی است که دکمه سمت راست ماوس در آن زمان فشرده بوده است.

نکته:

فسرده شدن همزمان چند دکمه از ماوس باعث می‌شود که مقدار پارامتر Button برابر جمع مقدار بر متناظر با دکمه‌های فشرده شده باشد. برای مثال در صورتی که هنگام وقوع رخداد مورد نظر، دکمه‌های چپ و راست ماوس فشرده شده باشند، مقدار متغیر Button برابر $3 = 1 + 2$ خواهد بود.

- پارامتر Shift: مقداری که این پارامتر در خود ذخیره می‌کند بیانگر این است که در هنگام وقوع رخداد کدام‌یک از کلیدهای Shift, Ctrl و یا Alt فشرده شده است. در واقع وضعیت این کلیدهای به صورت یک عدد Integer در متغیر Shift ذخیره می‌شوند.

شرح	مقدار	ثابت
وضعیت کلید Shift	1	vbShiftMask
وضعیت کلید Ctrl	2	vbCtrlMask
وضعیت کلید Alt	4	vbAltMask

برای مثال در صورتی که هنگام وقوع رخداد مقدار پارامتر Shift برابر عدد 2 باشد، بدین معنی است که کلید Ctrl در آن زمان فشرده بوده است.

نکته:

فسرده شدن همزمان چند کلید از مجموعه سه کلید مذکور باعث می‌شود که مقدار پارامتر shift برابر جمع مقادیر متناظر با کلیدهای فشرده شده باشد. برای مثال در صورتی که هنگام وقوع رخداد مورد نظر، هر سه کلید Shift, Ctrl و Alt فشرده شده باشند، مقدار متغیر Shift برابر $7 = 1+2+4$ خواهد بود.

نمونه:

برنامه‌ای بنویسید که با حرکت ماوس بر روی فرم در حالت فشرده بودن دکمه سمت چپ ماوس و کلید Shift، پیغام Good Luck! در جعبه پیغام ظاهر شود.

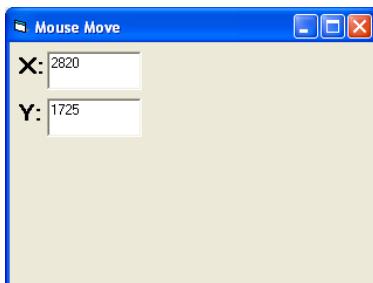
پاسخ:

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = 1 And Shift = 1 Then
    MsgBox "Good Luck!"
End If
End Sub
```

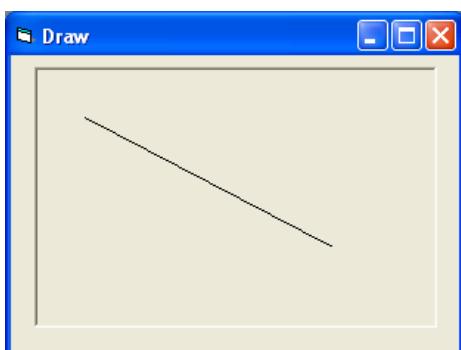
برنامه‌ی زیر نیز معادل برنامه‌ی نوشته شده در بالاست، با این تفاوت که در آن بهجای استفاده از اعداد مرتبط با دکمه‌ها و کلیدها، از ثابت‌ها استفاده کردایم.

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
As Single)
If Button = vbLeftButton And Shift = vbShiftMask Then
    MsgBox "Good Luck!"
End If
End Sub
```

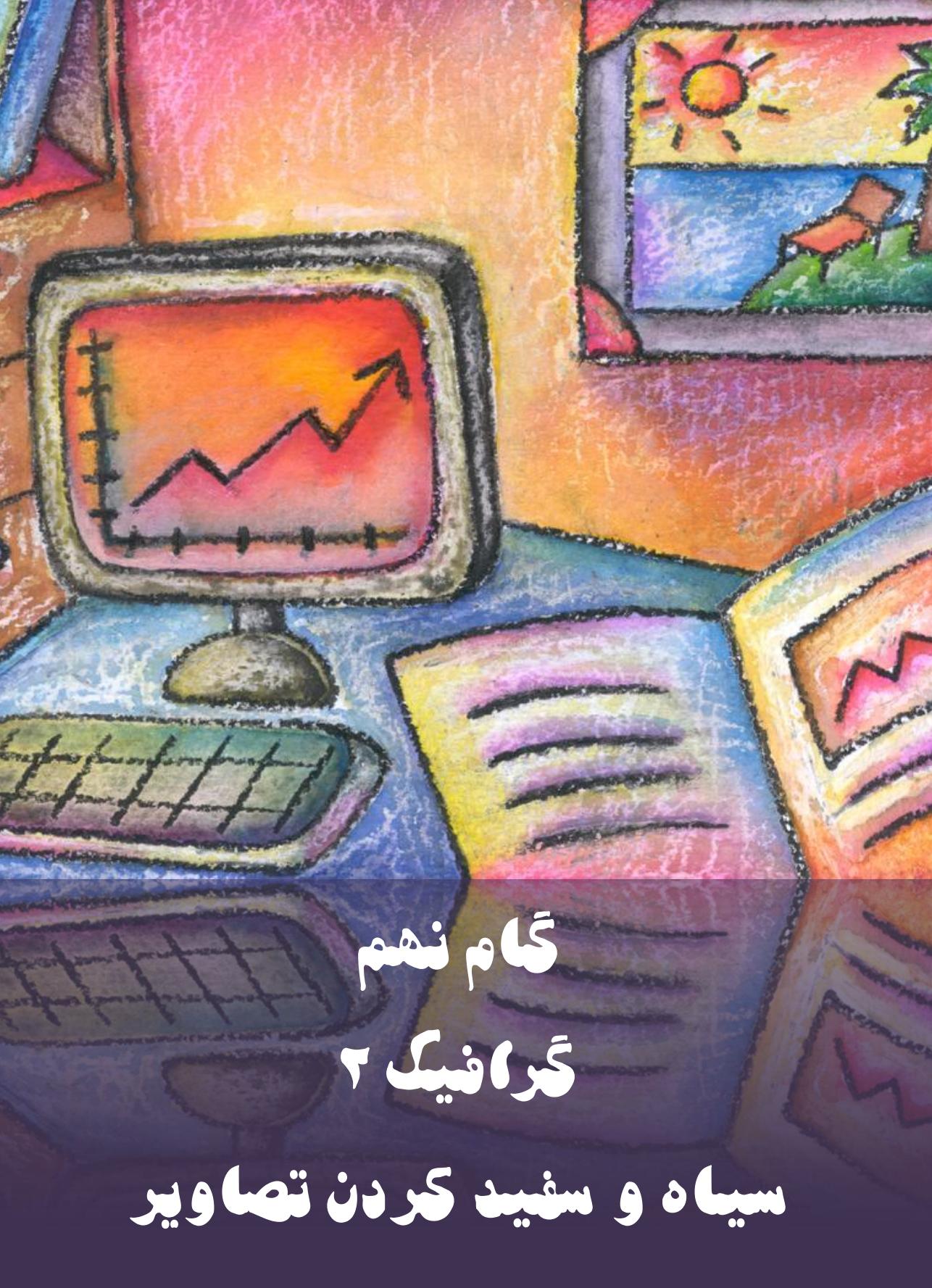
۵.۸. تمرین



۱. دو عدد جعبه متن (TextBox) بر روی فرم قرار دهید. برنامه‌ای بنویسید که مختصات حرکت ماوس بر روی فرم را در این دو جعبه متن (یکی برای مختصات X و دیگری برای مختصات Y) نشان دهد.



۲. برنامه‌ای بنویسید که در یک جعبه تصویر، خطی مابین نقاطی که دکمه سمت چپ ماوس فشرده و رها شده است، رسم نماید. در این تصویر دکمه ماوس در مکان نقطه‌ی بالا سمت چپ خط فشرده شده و در نقطه‌ی پایین سمت راست رها شده است.



گام نهم

گرافیک ۲

سیاه و سفید کردن تصاویر

۱.۹ آشنایی

در اکثر کارهای گرافیکی تبدیل تصاویر رنگی به سیاه و سفید کاربرد بسیاری دارد و با توجه به نوع و کاربرد تصویر، الگوریتم‌ها و فرمول‌های گوناگونی در این زمینه مطرح است. در جلسه امروز می‌خواهیم با بررسی یکی از این الگوریتم‌ها برنامه‌ای بنویسیم که عکسی را به حالت سیاه و سفید در آورد.

می‌دانیم که هر تصویر دارای تعدادی نقطه رنگی است و هر رنگ شماره‌ای دارد، برای مثال شماره‌ی رنگ قرمز خالص 255 و سفید خالص 16777215 و سیاه خالص 0 می‌باشد. همانطور که می‌دانید به خاطر سپردن این شماره‌ها در عمل غیر ممکن می‌باشد، بنابراین برای تولید این اعداد از روش‌های دیگری باید استفاده کرد. یکی از این روش‌ها استفاده از تابع RGB است که در جلسه ششم درباره آن صحبت کردیم. در این روش هر رنگ ترکیبی از سه رنگ پایه قرمز، سبز و آبی می‌باشد. تابع RGB مقدار این سه رنگ را که هر یک عددی بین 0 تا 255 است را گرفته و شماره رنگ حاصل از ترکیب این سه رنگ را می‌دهد. همان‌طور که قبلاً نیز توضیح داده شد، نحوه استفاده از تابع RGB به صورت زیر می‌باشد:

$$\text{Col} = \text{RGB}(r, g, b)$$

```
Pic_MyPic.Pset ( x, y ) , RGB ( 90, 150, 0 )
Frm_Main.BackColor = RGB(100,200,255)
```

به مثال‌های زیر توجه کنید:

شماره رنگ	مقدار آبی	مقدار سبز	مقدار قرمز
13145700	200	150	100
6592200	100	150	200
65535	0	255	255
255	0	0	255
65280	0	255	0
16711680	255	0	0
6579300	100	100	100
9868950	150	150	150
13158600	200	200	200
16777215	255	255	255

احتمالاً متوجه شده‌اید که رنگ‌های مابین سیاه و سفید (یعنی درجات مختلف خاکستری) دارای مقادیر مساوی در سه رنگ اصلی هستند. بنابراین برای تولید رنگ سیاه و سفید باید در تابع RGB مقادیر سه رنگ قرمز، آبی و سبز را مساوی یکدیگر قرار دهیم. حالا اگر بخواهیم یک تصویر رنگی را سیاه و سفید کنیم باید رنگ تک‌تک نقاط را بدست آورده و سپس مقادیر قرمز، آبی و سبز یعنی r , g , b ، آن رنگ را جدا کرده، از روی این سه مقدار یک مقدار مناسب ساخته و با استفاده از آن رنگ جدید را تولید نماییم. اگر این مقدار جدید را k بنامیم، رنگ جدید اینگونه ساخته می‌شود:

$$\text{RGB}(k, k, k)$$

یعنی مقدار هر سه رنگ اصلی مساوی است.

یک فرمول مناسب برای تبدیل r , g , b به مقدار مورد نظر که آن را K نامیدیم در زیر آمده است:

$$k = 0.3 * r + 0.59 * g + 0.11 * b$$

آیا می‌توانید علتی برای مقدار اعداد بالا بیابید؟

 نکته :

همانطور که قبلاً نیز اشاره شد، برای خواندن رنگ هر نقطه از تصویر موجود در `Pic_MyPic` از دستور `Point` استفاده کنید:

`Col = Pict_MyPic.Point(x, y)`

این تابع مقدار رنگ نقطه (x, y) را به صورت یک عدد برمی‌گرداند

برای جداسازی مقادیر r , g , b ابتدا نحوه کار تابع RGB را بررسی می‌کنیم. خروجی تابع RGB برابر فرمول‌های معادل زیر است:

$$RGB(r, g, b) = r \times 2^0 + g \times 2^8 + b \times 2^{16}$$

$$RGB(r, g, b) = r \times 256^0 + g \times 256^1 + b \times 256^2$$

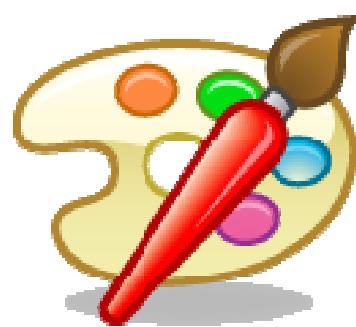
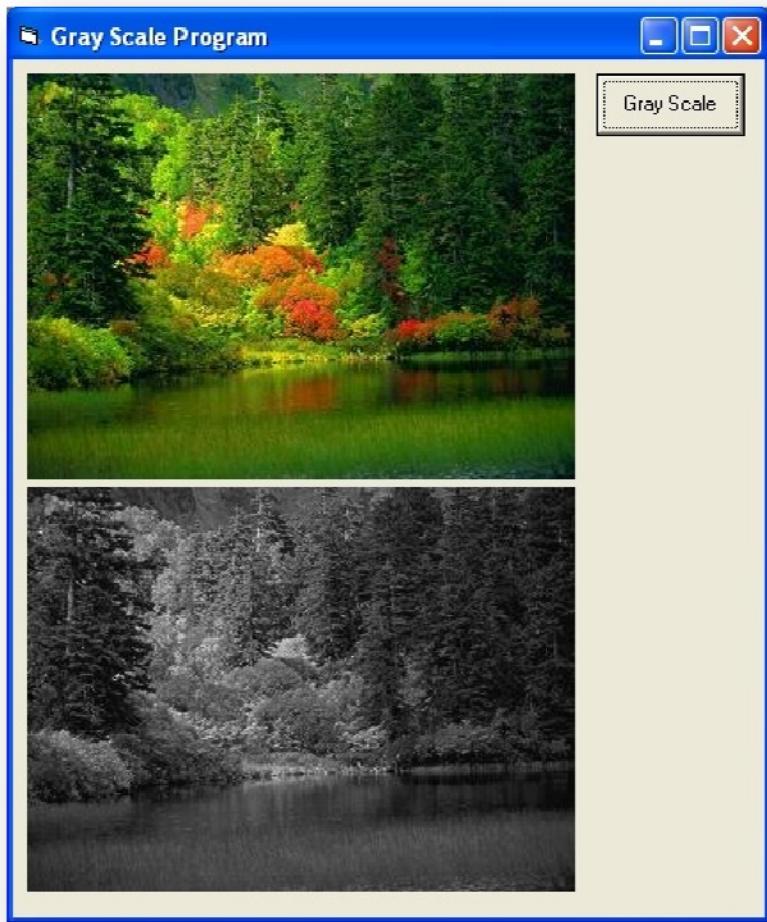
توجه کنید که فرمول‌های بالا هیچ چیز پیچیده‌ای نیستند و دقیقاً مانند ساختن یک عدد در مبنای ۱۰ می‌باشند:

$$123 = 3 \times 10^0 + 2 \times 10^1 + 3 \times 10^2$$

کاری که در واقع می‌خواستیم انجام دهیم، به دست آوردن مقادیر r, g, b از روی شماره رنگ بود. همان‌گونه که ارقام یک عدد ددهی (یک عدد در مبنای ۱۰) را جدا می‌کردیم، اکنون باید این روش جداسازی را بر مبنای ۲۵۶ انجام دهیم.

پس کار را به مراحل زیر تقسیم می‌کنیم تا به راحتی بتوانید آن‌ها را انجام دهید:

- **مرحله‌ی اول :** برنامه‌ای بنویسید که یک عدد را (از جعبه متن TextBox) گرفته و ارقام آن را در مبنای ۱۰ جدا نماید.
- **مرحله‌ی دوم :** برنامه‌ای بنویسید که یک شماره رنگ را گرفته r, g, b آن را جدا کند.
(جداسازی ارقام در مبنای ۲۵۶)
- **مرحله‌ی سوم :** برنامه‌ای ایجاد کنید که با توجه به توضیحات آمده در بالا یک تصویر را سیاه و سفید کند. (این کار با کلیک کردن بر روی دکمه فرمان Cmd_GrayScale انجام شود.) برنامه شما باید به شکل زیر باشد:



گام دهم

کار کردن با صفحه کلید

۱.۱۰. آشنایی

در این جلسه قصد داریم به نکاتی در رابطه با نحوه کار کردن با صفحه کلید پردازیم. همان طور که قبل‌آن نیز اشاره شد صفحه کلید به عنوان دستگاه ورودی، راهی است برای ارتباط بین کاربر و برنامه‌ها. با وجود جعبه‌متن و کنترل‌های گوناگون که برای ارتباط با کاربر پیش‌بینی شده‌اند، اما به کمک آن‌ها نمی‌توان تمامی رویدادهای صفحه کلید را کنترل نمود. سیستم‌عامل با هر ضربه‌ای که به صفحه کلید می‌خورد، رخدادهای مربوطه را تولید و به برنامه ارسال می‌کند.

۲.۱۰. رخدادهای صفحه کلید

رخدادهای اصلی مرتبط با صفحه کلید که در این جلسه به آن‌ها خواهیم پرداخت، عبارتنداز:

- .۱ KeyPress
- .۲ KeyDown
- .۳ KeyUp

همان‌طور که در جلسه کار کردن با ماوس دیدیم، هر رخداد وابسته به فرم و یا یک کنترل می‌باشد. رخدادهای فوق نیز در صورتی که کنترلی انتخاب و یا فوکوس داشته باشد، به آن وابسته است و در صورتی که هیچ کنترلی فوکوس نداشته باشد، رخداد مورد نظر به فرم می‌رسد.

۱.۲.۱۰. رخداد KeyPress

هرگاه کلیدی زده شود، این رخداد اتفاق می‌افتد.

در صورتی که بخواهیم برای وقوع رخداد Keypress در فرم، برنامه بنویسیم، دو خط آمده در زیر به طور خودکار ایجاد می‌شوند. اگر در قطعه برنامه زیر دقت کنید متوجه می‌شوید که یک عدد از نوع Integer پارامتر ورودی این زیر برنامه می‌باشد. این عدد در واقع کد اسکی کاراکتری است که کلید مربوط به آن بر روی صفحه کلید فشار داده شده است.

```
Private Sub Form_KeyPress(KeyAscii As Integer)
```

```
End Sub
```

۲.۲.۱۰. رخداد KeyDown

این رخداد زمانی اتفاق می‌افتد که کلیدی فشار داده باشد. (قبل از رها کردن دکمه رخداد صدای می‌شود).

شاید گمان کنید که وجود هر دو رخداد KeyDown و KeyPress لزومی نداشته باشد اما تفاوت‌های ظریفی بین این دو وجود دارد که در ادامه به چند مورد از آن‌ها اشاره خواهیم داشت اما قبل از آن لازم است به قطعه برنامه‌ای که ویژوال بیسیک به‌طور خودکار برای این رخداد ایجاد می‌کند، نگاهی داشته باشیم:

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
```

```
End Sub
```

همان‌طور که مشاهده می‌کنید این زیربرنامه دارای دو پارامتر ورودی به نام‌های KeyCode و Shift است. پارامتر KeyCode کد کلید پایین رفته را در خود ذخیره می‌کند و پارامتر Shift درست مشابه با پارامتر همنام که در مبحث ماوس با آن آشنا شدید رفتار می‌کند.

 نکته :

پارامتر Shift برای شناسایی کلیدهای Shift، Ctrl و Alt هم کاربرد دارد.

۳.۲.۱۰. تفاوت‌های بین دو رخداد KeyDown و KeyPress

- همان‌طور که می‌دانید کد اسکی حروف کوچک الفبا با حروف بزرگ متناظر تفاوت دارد. برای مثال کد اسکی A برابر 65 و کد اسکی a برابر 97 است. از آن‌جایی که پارامتر ورودی رخداد KeyPress کد اسکی را در خود ذخیره می‌کند، برای حروف کوچک و بزرگ اعداد متفاوتی در پارامتر KeyCode ذخیره می‌شود. اما پارامتر ورودی رخداد KeyAscii می‌شود. اما پارامتر ورودی رخداد KeyDown یعنی KeyCode

برای حروف کوچک و بزرگ (مانند A و a) کد یکسانی معادل با کد حرف بزرگ را ارائه می‌کند.

- مزیتی که رخداد KeyDown نسبت به KeyPress دارد، این است که با کمک آن می‌توان تقریباً تمام کلیدهای صفحه کلید را تشخیص داد. علاوه بر کلیدهای Alt و Ctrl و که توسط پارامتر ورودی Shift قابل شناسایی هستند، کلیدهایی نظیر End، Home، PageDown، PageUp، کلیدهای پیکانی و... که فاقد کد اسکی هستند همگی دارای کدی هستند که این کد در پارامتر KeyCode از رخداد KeyDown ذخیره می‌شود.

۴.۲.۱۰. KeyUp

هرگاه کاربر کلید فشرده شده‌ای را رها سازد، این رخداد اتفاق می‌افتد. ساختار و پارامترهای ورودی این رخداد مشابه با رخداد KeyDown می‌باشد.

 نکته :

به کمک رخدادهای KeyUp و KeyDown می‌توان برای فشرده و رها شدن یک کلید از صفحه کلید به طور جداگانه برنامه نوشت.

 نکته :

در صورتی که کلیدی را نمی‌دانید کافی است از یکی از برنامه‌های زیر استفاده نموده تا از کد کلید مورد نظر مطلع شوید.

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    MsgBox KeyCode
End Sub
```

```
Private Sub Form_KeyPress(KeyAscii As Integer)
    MsgBox KeyAscii
End Sub
```

مطمئن شوید که طرز کار دو قطعه برنامه فوق را می‌دانید.

۳.۱۰. تمرین

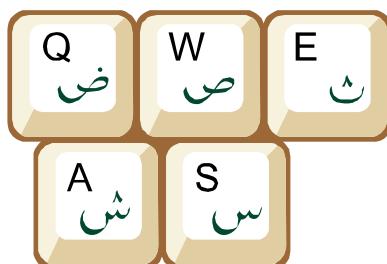
۱. یک جعبه متن (TextBox) بر روی فرم قرار داده، با کمک رخدادهای فوق برنامه‌ای بنویسید که اگر کلمه عبور "VisualBasic" در جعبه متن وارد شده بود و کاربر کلیدهای Ctrl+Enter را بزند، برنامه پیغام دهد "Access Granted" و در غیر این صورت پیغام دهد "Access Denied".
۲. برنامه‌ای بنویسید که با فشار دادن و رها شدن کلیدهای مشخص شده در جدول زیر کارهای خواسته شده را انجام دهد:

کاری که باید انجام شود	کلید
رنگ زمینه* فرم به رنگ قرمز در آید.	فشردن شدن کلید R یا r
پیغام "Welcome" ظاهر شود.	زده شدن کلید Enter
دوبار زده شدن پیاپی کلید Shift	طول و عرض** فرم به اندازه 10 واحد افزایش یابد.
فرم به در بالای صفحه نمایش قرار بگیرد.	Rها شدن کلید PageUp ***.

* از خصوصیت BackColor استفاده نمایید.

** از خصوصیت‌های Weight و Height استفاده کنید.

*** از خصوصیت Top استفاده کنید.



گام یازدهم

گرافیک ۲

بزرگ نمایی تصاویر

۱.۱۱ آشنایی

در اکثر کارهای گرافیکی، بزرگ و کوچک کردن تصاویر کاربردهای بسیاری دارد و در کارهای تخصصی و غیرتخصصی مورد استفاده اکثر کاربران قرار می‌گیرد. در این جلسه به بزرگنمایی تصاویر گرافیکی می‌پردازیم. یکی از ساده‌ترین الگوریتم‌های بزرگنمایی به صورت زیر عمل می‌کند:

هر تصویر دارای تعدادی نقطه است. برای مثال اگر به ازای هر نقطه، 4×4 نقطه با همان رنگ نقطه اولیه قرار داده شود (یعنی یک مربع 2×2)، در هر بعد بزرگنمایی 2 برابر خواهیم داشت. پس الگوریتم کلی کار را می‌توان در جمله زیر خلاصه کرد:

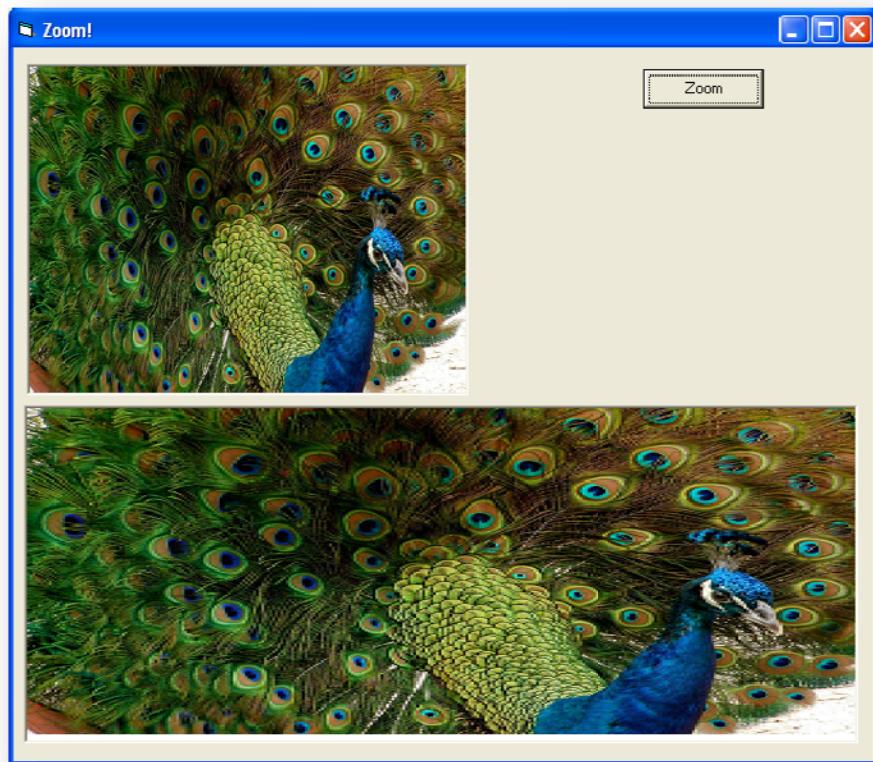
قرار دادن مجموعه‌ای مربعی از نقاط به ازای هر نقطه با همان رنگ نقطه اولیه.

کار را به مراحل زیر تقسیم می‌کنیم تا بتوانیم به سادگی آن را انجام دهیم:

- **اول :** برنامه‌ای بنویسید که یک تصویر را در بعد طولی 2 برابر بزرگ نماید. به شکل صفحه‌ی بعد توجه کنید.

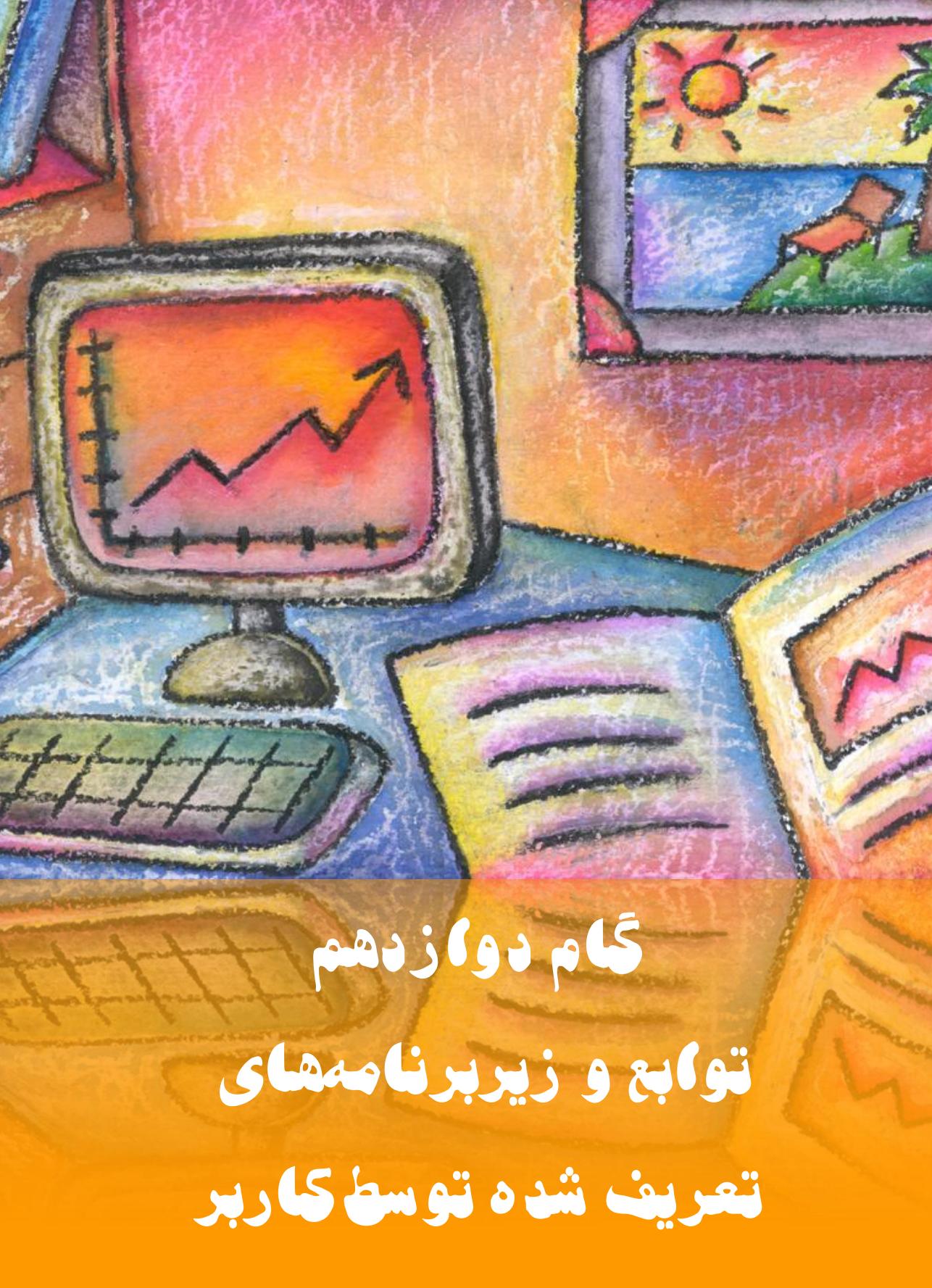
- **دوم :** برنامه‌ای بنویسید که یک تصویر را در دو بعد طولی و عرضی به اندازه‌ی 2 برابر بزرگ نماید.

- **سوم :** برنامه‌ای بنویسید که یک تصویر را در دو بعد طولی و عرضش به اندازه دلخواه k برابر بزرگ نماید. عدد k را می‌توانید به کمک یک جعبه‌ی متن (TextBox) از کاربر دریافت کنید.



۲.۱۱. تمرین

۱. بزرگنمایی واقعی: برنامه‌ای بنویسید که بخش دلخواهی از تصویر مورد نظر را به اندازه دلخواه k برابر بزرگ کند. عدد k را می‌توانید به کمک یک جعبه‌متن (TextBox) از کاربر دریافت نمایید. همچنین کاربر برای انتخاب بخش مورد نظر از تصویر برای عمل بزرگنمایی، از ماوس استفاده خواهد کرد.
۲. کوچکنمایی: برنامه‌ای بنویسید که تصویر مورد نظر را به اندازه‌ی دلخواه کوچک کند.



گام دوازدهم

توابع و زیربرنامه‌های

تعریف شده توسط کاربر

۱.۱۲. آشنایی

تاکنون با نمونه‌هایی از زیربرنامه آشنا شده‌اید. برای مثال زمانی که برای رخداد Click دکمه فرمان برنامه می‌نوشتید در واقع زیربرنامه‌ای را کامل می‌کردید که قسمتی از آن را ویژوال بیسیک قبل نوشته بود.

```
Private Sub Cmd_Sum_Click()
```

```
End Sub
```

زمانی که قصد دارید برای Click دکمه فرمان برنامه بنویسید، ویژوال بیسیک دو خط ابتدا و انتهای زیربرنامه را تولید و شما باید داخل آن را بنا به مساله مورد نظر کامل کنید.

اگر به برنامه‌هایی که امسال با ویژوال بیسیک نوشته‌اید دقت کنید، به تفاوت بین آن‌ها با برنامه‌های بیسیک پی خواهید برد. اشتباه نکنید! منظور ما از تفاوت چیزی غیر از ویژوال (گرافیکی) بودن برنامه‌هاست.

برنامه‌هایی که در سال گذشته به کمک زبان بیسیک می‌نوشتید، معمولاً شامل برنامه‌ای بود که از نقطه‌ای آغاز و در نقطه‌ای به پایان می‌رسید. اما برنامه‌هایی که در این چند جلسه در محیط ویژوال بیسیک تولید کردید از تعدادی زیربرنامه تشکیل شده است.

تابع و زیربرنامه هر دو قطعه برنامه‌هایی هستند که برای اجرای کاری مشخص نوشته می‌شوند. همچنین با نامی که به آن‌ها اختصاص داده می‌شود، می‌توان آن‌ها را فرخوانی و یا به عبارت دیگر اجرا نمود. در سال گذشته با تعداد زیادی تابع آشنا شده‌اید اما شاید در آن موقع نمی‌دانستید که با چه چیزی طرف هستید. نمونه‌هایی از تابع‌هایی که در زبان بیسیک وجود دارد را در زیر آورده‌ایم (تعدادی زیادی از توابعی که در زبان بیسیک وجود دارد، معادلی در زبان ویژوال بیسیک نیز دارند):

- Int()
- Abs()
- Mid\$()
- Left\$()

مثال‌هایی که در بالا آورده شد، در واقع تابع‌هایی هستند که توسط طراحان زبان بیسیک برای استفاده کاربران طراحی و نوشته شده‌اند. پس از این جلسه می‌توانیم تابع و زیربرنامه‌های مورد نیاز را حتی اگر از قبل در زبان برنامه‌نویسی وجود نداشته باشد، بنویسیم. در این جلسه به موضوعات زیر خواهیم پرداخت:

- تفاوت‌های بین تابع و زیربرنامه
- نحوه تعریف تابع و زیرنامه
- چگونگی استفاده از تابع و زیربرنامه تعریف شده

۲.۱۲. تفاوت‌های بین تابع و زیربرنامه

قبل از وارد شدن به جزئیات کار خوب است به دو مثال زیر توجه داشته باشید:

- فرض کنید تابع `(Abs()` در زبان ویژوال بیسیک وجود نداشت و ما خودمان می‌خواستیم چنین تابعی را تعریف کنیم به نحوی که یک عدد از نوع `Long` را گرفته و قدرمطلق آن را به عنوان خروجی برگرداند.

آیا می‌دانید قدر مطلق یعنی چه؟

قدر مطلق یک عدد برابر است با مقدار آن عدد بدون در نظر گرفتن علامت. برای مثال قدر مطلق عدد -2 برابر است با 2 و یا قدر مطلق عدد $\frac{3}{5}$ برابر با $\frac{3}{5}$ می‌باشد. در واقع اگر عدد منفی باشد، برای بدست آوردن قدر مطلق کافی است آن را در یک منفی ضرب نموده و اگر عدد مثبت باشد کاری خاصی نباید انجام شود

Public Function Abs(Number As Long) As Long

```
If Number < 0 Then
    Abs = (-1) * Number
Else
    Abs = Number
End If

End Function
```

- می‌خواهیم زیربرنامه‌ای بنویسیم که با گرفتن مدت زمان انجام کاری بر حسب ساعت، دقیقه و ثانیه، آن را بر حسب ثانیه محاسبه و نتیجه را به صورت یک جعبه پیغام (MsgBox) نشان دهد.

Private Sub Time_Duration(Hours As Integer, Minutes As Integer, Seconds As Integer)

```
Dim D As Long
D = 3600 * Hours + 60 * Minutes + Seconds

MsgBox D

End Sub
```

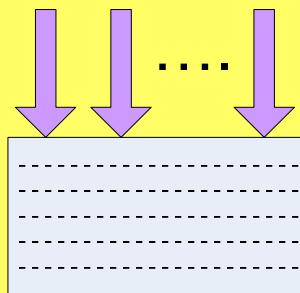
قبل از آن که به چگونگی تعریف و نحوه استفاده از تابع و زیربرنامه پردازیم، لازم است با تفاوت بین این دو مفهوم به طور دقیق‌تر آشنا شویم.

زیربرنامه:

زیربرنامه در واقع قطعه برنامه‌ی دارای نامی است که برای انجام کاری نوشته می‌شود. در موقع نیاز می‌توان با فراخوانی زیربرنامه به کمک نام آن باعث اجرای کار تعریف شده در آن شد.

زیر برنامه با دریافت تعدادی پارامتر ورودی، کاری را بر اساس آن پارامترها انجام خواهد داد. البته توجه به این نکته ضروری است که تعداد پارامترهای ورودی یک زیربرنامه می‌تواند صفر باشد و به عبارت دیگر زیربرنامه پارامتر ورودی نداشته باشد.

در مثال قبل (Time_Duration) زیربرنامه با دریافت ۳ پارامتر ورودی محاسبات لازم را انجام داده و نتیجه را به صورت جعبه پیغام نمایش می‌دهد.

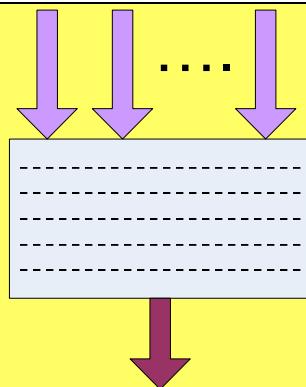


تابع

تابع قطعه برنامه‌ی دارای نامی است که برای انجام کاری نوشته می‌شود یعنی مشابه با زیربرنامه. تنها تفاوت تابع با زیربرنامه در این است که تابع مقداری (عددی، حرفی و...) را به عنوان خروجی برمی‌گرداند. برای استفاده از تابع در موقع نیاز، می‌توان آن‌ها فراخوانی نمود.

تابع با دریافت تعدادی پارامتر ورودی، کاری را بر اساس آن پارامترها انجام داده و همچنین مقداری را به عنوان خروجی برمی‌گرداند. توجه کنید که تعداد پارامترهای ورودی یک تابع می‌تواند صفر باشد و به عبارت دیگر پارامتر ورودی نداشته باشد، که البته این حالت، کاربرد زیادی ندارد.

در مثال قبل (Abs) تابع با دریافت یک عدد به عنوان ورودی، قدرمطلق آن را محاسبه و به عنوان خروجی برمی‌گرداند.



۳.۱۲. نحوه تعریف زیربرنامه

زیربرنامه در ویژوال بیسیک به صورت زیر تعریف می‌شود.

[Private | Public] Sub [نام زیربرنامه] (فهرست پارامترها)

...

...

End Sub

به نکات زیر در رابطه با نحوه تعریف زیربرنامه توجه کنید:

- حداکثر یکی از دو عبارت Private و یا Public می‌تواند قبل از عبارت Sub قرار بگیرد. همان طور که قبلاً با دو مفهوم Private و Public آشنا شدید، اگر بخواهیم که زیربرنامه تنها در فرم (و یا مازولی) که در آن تعریف شده است، قابل استفاده باشد از عبارت Private به معنای خصوصی استفاده می‌کنیم و اگر بخواهیم زیربرنامه در تمامی فرم‌ها (و مازول‌ها) قابل استفاده باشد از عبارت Public استفاده خواهیم کرد.
- نام زیربرنامه به دلخواه برنامه‌نویس است. اما استفاده از بعضی کلمات مانند "Name" برای نام زیربرنامه مجاز نمی‌باشد. در صورت بروز چنین خطایی برای نام زیربرنامه باید نام آن را تغییر دهید. پیشنهاد می‌کنیم که از نام‌های بامعنای برای نام‌گذاری زیربرنامه‌ها استفاده نمایید.

- زیربرنامه می‌تواند دارای تعدادی پارامتر ورودی باشد که در هنگام فراخوانی مقادیری به آن‌ها اختصاص می‌دهیم. در فهرست پارامترها نام و نوع پارامترها معرفی شده و در صورتی که بیش از یک پارامتر داشته باشیم، بین آن‌ها علامت ویرگول قرار می‌دهیم.
بهتر است بار دیگر نگاهی بر زیربرنامه تعریف شده در بالا یعنی Time_Duration داشته باشیم:

```
Private Sub Time_Duration(Hours As Integer, Minutes As Integer,
                           Seconds As Integer)

    Dim D As Long
    D = 3600 * Hours + 60 * Minutes + Seconds
    MsgBox D
End Sub
```

} بدنی زیر برنامه

همان‌طور که ملاحظه می‌کنید برای تعریف این زیربرنامه از عبارت Private استفاده شده است، پس بدین معنی است که این زیربرنامه تنها در فرمی که در آن تعریف شده است قابل استفاده می‌باشد. همچنین نام زیربرنامه برابر Time_Duration قرار داده شده است. این زیربرنامه دارای سه پارامتر ورودی به نام‌های Minutes و Seconds و Hours می‌باشد. در داخل زیربرنامه از این سه پارامتر ورودی استفاده شده است تا کار مورد نظر در داخل بدنی زیربرنامه به درستی انجام شود. نکته‌ی مهم و لازم به توجه این است که نام پارامترهای ورودی یک زیربرنامه در بدنی آن شناخته شده و قابل استفاده‌اند.

۴.۱۲. نحوه فراخوانی و یا استفاده از زیربرنامه

برای فراخوانی و یا استفاده از زیربرنامه دو روش زیر امکان‌پذیر است:

۱. در روش اول برای استفاده از زیربرنامه کافی است ابتدا عبارت Call و سپس نام زیربرنامه را در محل مورد نظر بنویسیم. در صورتی که زیربرنامه دارای پارامتر ورودی است، باید مقادیر

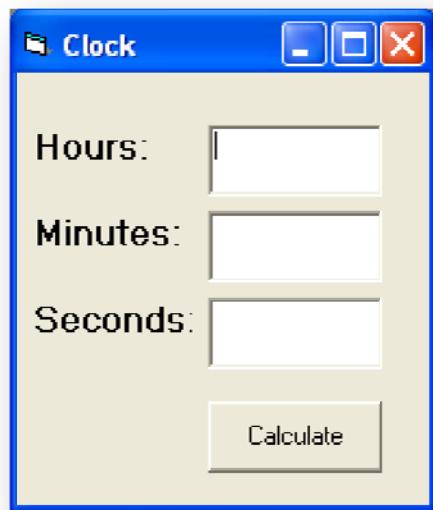
پارامترهای ورودی را به ترتیب درست و در داخل پرانتز جلوی نام زیربرنامه قرار دهیم. به مثال زیر توجه کنید:

Call Time_Duration (Val(Txt_Hours.Text), Val(Txt_Minutes.Text),
Val(Txt_Seconds.Text))

۲. روش دیگر برای فراخوانی زیربرنامه این است که نام زیربرنامه را بدون عبارت Call در محل مورد نظر آورده و در ادامه بدون استفاده از علامت پرانتز، مقادیر پارامترهای ورودی را بیاوریم.
به مثال زیر توجه کنید:

Time_Duration Val(Txt_Hours.Text), Val(Txt_Minutes.Text),
Val(Txt_Seconds.Text)

فرض کنید که سه جعبه‌ی متن (TextBox) بر روی فرم قرار داده شده است. یکی برای ورود عدد ساعت، یکی برای عدد دقیقه و آخری نیز برای ورود عدد ثانیه. همچنین یک دکمه‌ی فرمان (CommandButton) نیز بر روی فرم قرار دارد که قصد داریم با کلیک کردن بر روی آن تبدیل یافته سه مقدار دورن جعبه‌های متن بر حسب ثانیه نمایش داده شود. به برنامه‌های زیر توجه کنید.



```
Private Sub Time_Duration(Hours As Integer, Minutes As Integer, Seconds
As Integer)
```

```
Dim D As Long
```

```
D = 3600 * Hours + 60 * Minutes + Seconds
```

```
MsgBox D
```

```
End Sub
```

```
Private Sub Cmd_Calculate_Click()
```

```
    Time_Duration Val(Txt_Hours.Text), Val(Txt_Minutes.Text),
    Val(Txt_Seconds.Text)
```

```
End Sub
```

همان‌طور که مشاهده می‌کنید در رخداد کلیک مربوط به دکمه فرمان تنها کافی است که زیربرنامه مورد نظر را فراخوانی کنیم. برنامه توضیح داده شده در بالا را آزمایش نمایید.

۵.۱۲. نحوه تعریف تابع

همان‌طور که در ابتدای نیز توضیح داده شد، تابع زیربرنامه‌ای است که مقدار (عددی، حرفی و...) به عنوان خروجی برمی‌گرداند. تابع در ویندوز بیسیک به صورت زیر تعریف می‌شود.

```
[Private | Public] Function نام_تابع As (فهرست پارامترها)
```

```
...
```

```
        عبارت_یا_مقدار_خروجی = نام_تابع
```

```
...
```

```
End Function
```

به نکات زیر در رابطه با نحوه تعریف تابع توجه کنید:

- همانند زیربرنامه، حداکثر یکی از دو عبارت **Public** و یا **Private** می‌تواند قبل از عبارت **Function** قرار بگیرد. عملکرد این دو نیز مشابه حالت قبل است.
- نام تابع به دلخواه برنامه‌نویس است. اما استفاده از بعض کلمات برای نام تابع مجاز نمی‌باشد. پیشنهاد می‌کنیم که از نام‌های بامعنای برای نام‌گذاری تابع استفاده نمایید.
- تابع می‌تواند دارای تعدادی پارامتر ورودی باشد که در هنگام فراخوانی مقداری به آن‌ها اختصاص می‌دهیم. در فهرست پارامترها نام و نوع پارامترها معرفی شده و در صورتی که بیش از یک پارامتر داشته باشیم، بین آن‌ها علامت ویرگول قرار می‌دهیم.
- هر تابع دارای یک خروجی (از نوع عددی، حرفی و...) است. در تعریف تابع باید نوع خروجی (برای مثال: **String**, **Single**, **Integer** و...) آن را مشخص نماییم. برای این کار کافی است که نوع خروجی را به همراه عبارت **As** بعد از پرانتز مربوط به فهرست پارامترها بیاوریم.
- مقداری که تابع به عنوان خروجی برمی‌گرداند باید از نوعی باشد که در خط اول آن تعریف شده است. برای مثال اگر نوع خروجی تابع را از نوع **Integer** تعریف کرده‌اید، دیگر نمی‌توانید مقداری حرفی را به عنوان خروجی برگردانید. برای این که مقدار خروجی تابع را مشخص کافی است نام تابع را برابر مقدار و یا عبارت خروجی مورد نظر قرار دهید. برای روشن‌تر شدن موضوع به نحوه تعریف تابع در بالا دقت کنید.

بار دیگر نگاهی بر تابع **Abs** تعریف شده در بالا خواهیم داشت:

Private Function Abs(Number As Long) As Long

```
If Number < 0 Then
    Abs = (-1) * Number
Else
    Abs = Number
End If
End Function
```

بدنه‌ی تابع

همان‌طور که ملاحظه می‌کنید برای تعریف این تابع از عبارت **Public** استفاده شده است، بدین معنی که این تابع در تمامی فرم‌ها (و مازول‌ها) قابل استفاده می‌باشد. نام تابع برابر **Abs** قرار داده شده است. این تابع دارای یک پارامتر ورودی به نام **Number** و از نوع **Long** است. همچنین نوع خروجی تابع نیز **Long** می‌باشد. در داخل بدن تابع از این پارامتر ورودی استفاده شده است تا کار مورد نظر یعنی محاسبه مقدار قدرمطلق پارامتر ورودی به درستی انجام شود. توجه کنید که نام پارامترهای ورودی یک تابع در بدن آن شناخته شده و قابل استفاده‌اند. اگر در مثال بالا دقت کنید در دو خط سوم و پنجم، نام تابع برابر با عبارت مورد نظر برای خروجی قرار گرفته است. با توجه به ساختار **If** که در بدن تابع وجود دارد، در صورتی که شرط **0 < Number** برقرار باشد، نام تابع برابر ***(-1)** و در غیر این صورت برابر مقدار **Number** خواهد بود.

۶.۱۲. نحوه فراخوانی و یا استفاده از تابع

برای فراخوانی و یا استفاده از تابع به روش زیر عمل می‌کنیم. کافی است نام تابع را در محل مورد نظر نوشته و مقادیر پارامترهای ورودی را به ترتیب درست و در داخل پرانتز جلوی نام تابع قرار دهیم. به دلیل این‌که تابع دارای مقداری به عنوان خروجی است، فراخوانی آن با فراخوانی زیربرنامه‌ها اندکی متفاوت است. به مثال‌های زیر دقت کنید:

```
Answer = Abs(-10)
Result = Abs(Number)
Print Abs(-100)
Msgbox Abs(10)
Txt_Result.Text = Str(Abs(5))
```

۷.۱۲. تمرین

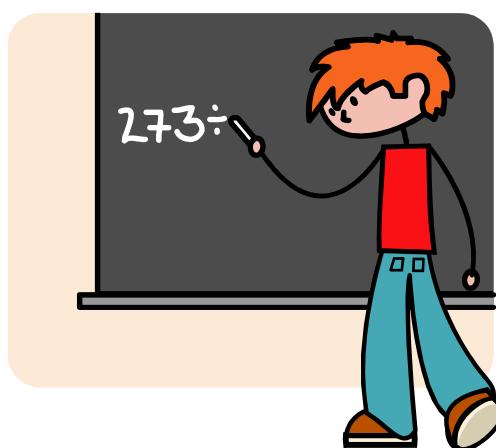
۱. تابعی بنویسید که یک عدد صحیح به عنوان ورودی گرفته و فاکتوریل آن را به عنوان خروجی برگرداند.

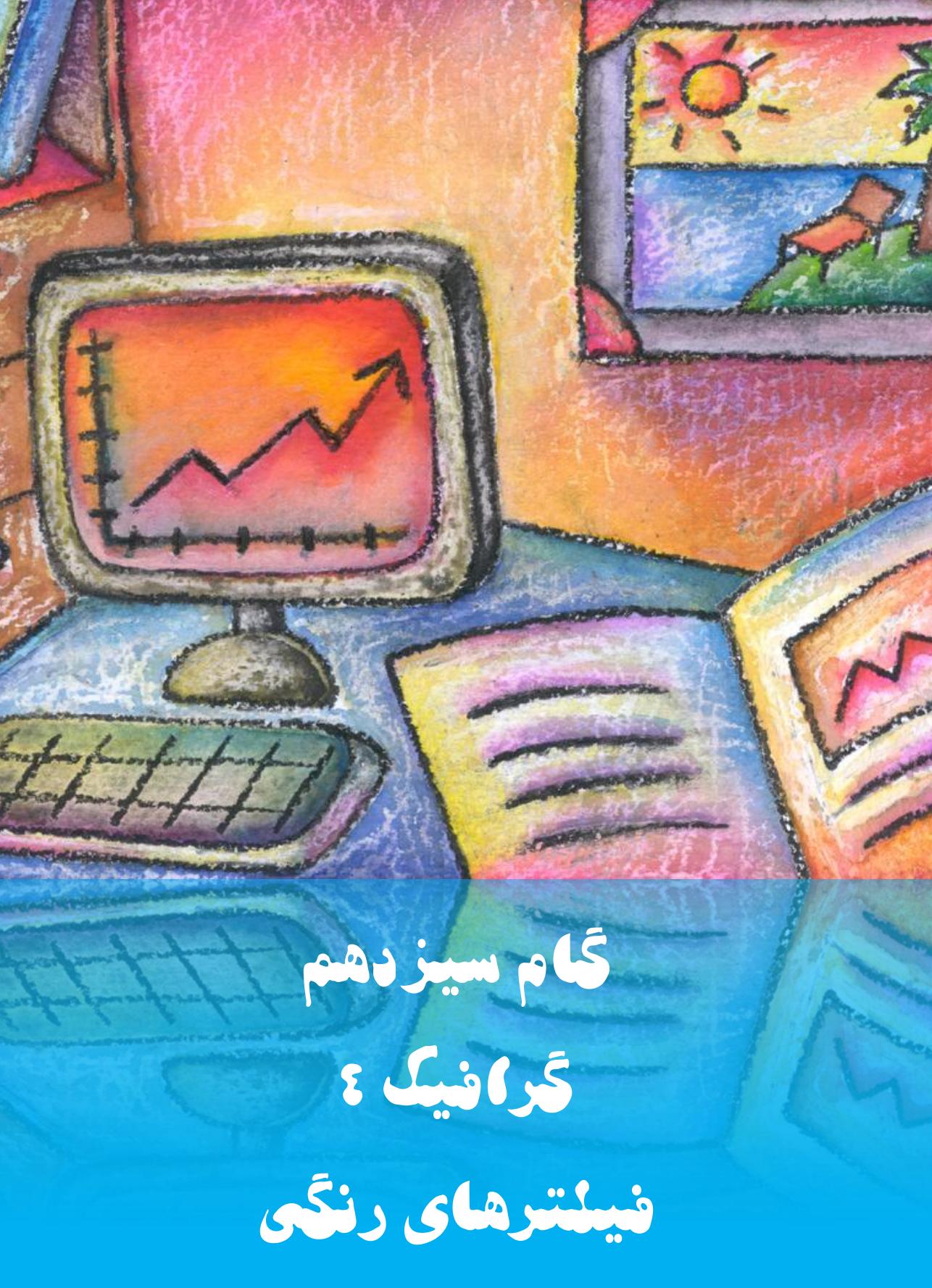
۲. تابعی بنویسید که یک عدد اعشاری به عنوان ورودی گرفته و مقدار گرد شده آن را برگرداند.

۳. تابعی بنویسید که دو رشته به عنوان ورودی گرفته و بررسی کند که آیا رشته اول شامل رشته دوم است یا خیر؟ در صورت مثبت بودن جواب، تابع مقدار True و در غیراین صورت مقدار False برگرداند.

راهنمایی: در تعریف تابع، نوع خروجی را از نوع Boolean در نظر بگیرید.

۴. تابعی بنویسید که یک شماره رنگ را به عنوان ورودی گرفته و مقدار رنگ قرمز آن را برگرداند. این کار را می‌توانید برای رنگ‌های سبز و آبی نیز انجام دهید.





گام سیزدهم

گرافیک ؛

فیکرهای رنگی

۱.۱۳ آشنایی

در جلسه امروز قصد داریم تا با تعدادی از فیلترهای گرافیکی و رنگی آشنا شویم؛ امکانی که در بسیاری از برنامه‌های گرافیکی وجود دارد. در بسیاری از اعمال گرافیکی، بایستی بر روی رنگ نقاط تصویر، عملیات ریاضی انجام بگیرد. در اکثر موارد، برای اعمال تغییرات برای تصاویر گرافیکی بایستی ابتدا مقادیر رنگ‌های قرمز، سبز و آبی رنگ جدا شده و سپس بر روی هر کدام این عمل مورد صورت گیرد، برای مثال در مورد میانگین‌گیری داریم:

$$R = \frac{R_1 + R_2 + R_3 + \cdots + R_N}{N}$$

$$G = \frac{G_1 + G_2 + G_3 + \cdots + G_N}{N}$$

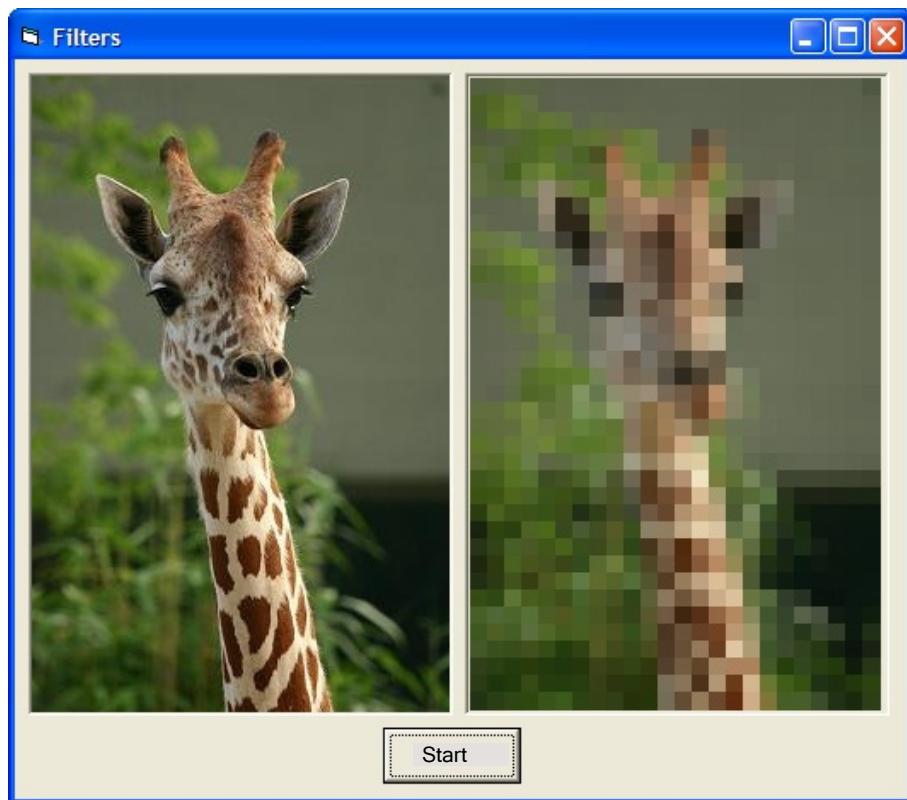
$$B = \frac{B_1 + B_2 + B_3 + \cdots + B_N}{N}$$

که در آن از مقادیر r و g و b رنگ N نقطه میانگین گرفته شده است و یک رنگ جدید به دست آمده است.

۲.۱۳. فیلتر اول: شطرنجی کردن تصویر

برنامه‌ای بنویسید که یک تصویر را به کمک مربع‌های 10×10 شطرنجی کند، یعنی از رنگ تمام نقاط در مربع‌هایی به ابعاد ده پیکسل در ده پیکسل از تصویر اصلی میانگین گرفته و در تصویر نهایی یک مربع با رنگ میانگین رسم کند.

برای روشن‌تر شدن موضوع به تصویر صفحه‌ی بعد دقت کنید:



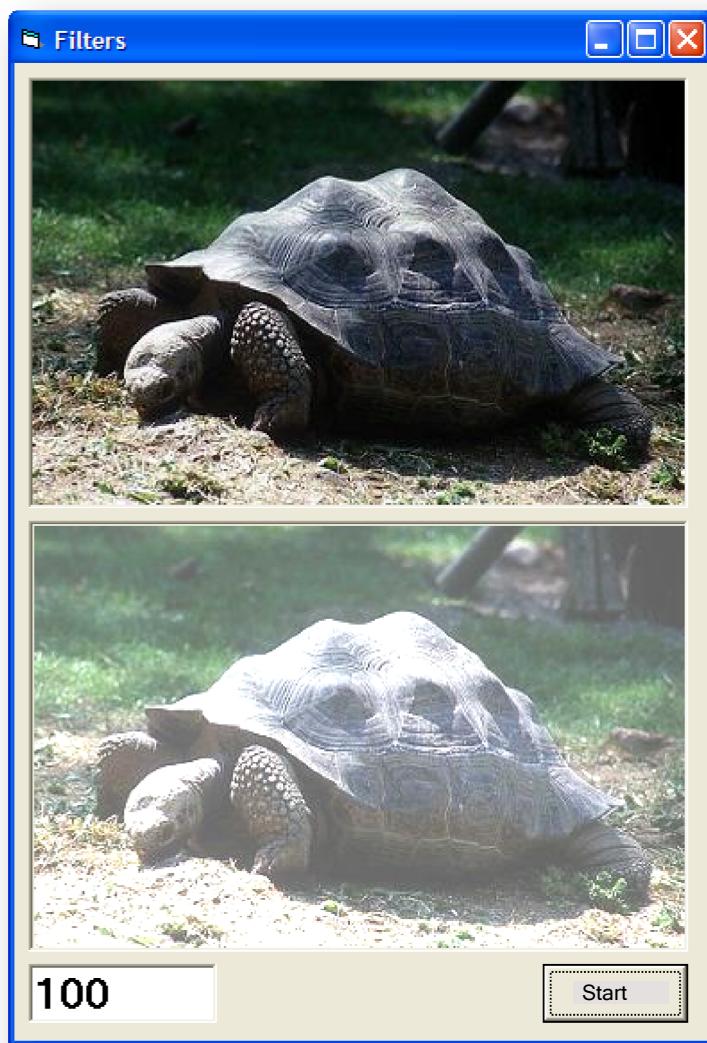
عکس سمت چپ بدون تغییر و عکس سمت راست با اعمال فیلتر است.

۳.۱۳. فیلتر دوم: تنظیم روشنایی تصویر

برنامه‌ای بنویسید که بتوان به کمک آن میزان روشنایی تصویر را کنترل کرد. میزان روشنایی تصویر و یا یک نقطه در واقع میزان سفید بودن آن است. همان‌طور که می‌دانید، مقادیر سبز، قرمز و آبی رنگ سفید برابر 255 و برای رنگ سیاه برابر 0 است. برای تغییر میزان سفید بودن بایستی سه مقدار قرمز، سبز و آبی رنگ را به یک میزان کم یا زیاد کنید. **دقت کنید که در هنگام تغییر مقدار رنگ‌ها، اگر این مقادیر منفی شدند، مقدار آن‌ها را برابر صفر و اگر بزرگتر از 255 شدند، مقدار آن‌ها را برابر 255 قرار دهید.**

برای گرفتن مقدار نظر می‌توانید از کنترل جعبه‌ی متن استفاده کنید.

تمرین: اگر بخواهید برای دریافت این مقدار از کنترل ScrollBar استفاده کنید، چه کاری باید بکنید؟



۴.۱۳. فیلتر سوم: ترکیب دو تصویر

برنامه‌ای بنویسید که دو تصویر را با هم ترکیب کند. برای این کار کافی است میانگین رنگ یک نقطه خاص را از دو تصویر مورد نظر حساب کرده و نقطه‌ای با رنگ میانگین در تصویر سوم رسم نمایید.



گام چهاردهم کارکردن با فایلها



۱.۱۴. آشنایی

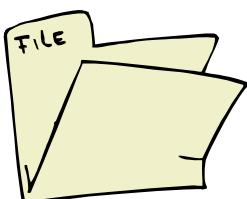
همان‌طور که می‌دانید حافظه اصلی کامپیوتر موقتی محسوب شده و با قطع برق اطلاعات آن از بین می‌رود. از طرف دیگر بسیاری از برنامه‌ها نیاز به ذخیره کردن دائمی اطلاعات خود داشته، بدین ترتیب که با قطع برق و یا روشن و خاموش شدن کامپیوتر بتوانند از اطلاعات قبلی خود استفاده نمایند. برنامه‌ها برای ذخیره کردن دائمی اطلاعات از فایل‌ها استفاده می‌کنند.

فایل‌ها بر دو نوع هستند: فایل‌های متنی و فایل‌های باینری. در فایل‌های متنی، فقط کاراکترهای اسکی و تعدادی کاراکتر خاص مانند کاراکتر انتهای خط و کاراکتر انتهای فایل وجود دارند. یعنی هر بایت از این فایل نمی‌تواند تمام مقادیر ما بین ۰ تا ۲۵۵ را قبول کند. فایل‌های با پسوند TXT، HTM و BAS جز این دسته فایل‌ها محسوب می‌شوند. از طرف دیگر، فایل‌های باینری می‌توانند حاوی تمامی کاراکترهای ممکن باشند. فایل‌های باینری کاربردهای گسترده‌ای دارند، برای مثال فایل‌های تصویری، صوتی، فیلم، برنامه‌ها و... از نوع فایل‌های باینری هستند. می‌توان این‌گونه نتیجه گرفت که فایل‌های متنی، زیرمجموعه‌ای از فایل‌های باینری محسوب می‌شوند.

۲.۱۴. فایل‌های متنی

همان‌طور که در بالا توضیح داده شد، داده‌های متنی را می‌توان در فایل‌های متنی ذخیره نمود و برای ذخیره کردن داده‌های پیچیده‌تر، مانند داده‌های صوتی و تصویری باید از فایل‌های باینری استفاده کرد.

باید توجه کنید که فایل‌های متنی ترتیبی هستند؛ یعنی برای این که خط N ام فایل را بخوانید، ابتدا باید N-1 خط قبلی را خوانده باشید. در ادامه به نحوه کار کردن با فایل‌های متنی خواهیم پرداخت.



۳.۱۴. نحوه کار کردن با فایل‌های متنی در ویژوال بیسیک

برای کار کردن با فایل‌ها، همواره سه مرحله‌ی زیر را به خاطر داشته باشید:

۱. باز کردن فایل (با کمک دستور **Open**)
۲. خواندن داده‌ها از فایل (با کمک دستور **#Line Input** و یا دستور **#Input**) و یا نوشتن داده‌ها در فایل (با کمک دستور **#Print**)
۳. بستن فایل (با کمک دستور **Close**)

حال باید دید که چگونه می‌توان با کمک دستورات بالا، اطلاعاتی را از یک فایل متنی خواند و یا در یک فایل متنی ذخیره کرد.

۱.۳.۱۴. باز کردن فایل با دستور **OPEN**

همان‌طور که در بالا اشاره کردیم از دستور **Open** برای باز کردن فایل مورد نظر استفاده می‌کنیم. باید توجه کنید که قبل از این‌که بتوان اطلاعاتی را از فایل خواند و یا در فایل نوشت، باید آن فایل را باز نمود. دستور **Open** دارای پارامترهای زیر می‌باشد:

Open	شماره‌ی فایل #	For	نام فایل	As	حالت فایل
-------------	----------------	------------	----------	-----------	-----------

- نام فایل: یک عبارت رشته‌ای و یا متغیر رشته‌ای است که نام فایل مورد نظر به همراه مسیر فایل در آن وجود دارد.
- حالت باز کردن فایل: تعیین‌کننده این است که می‌خواهیم فایل را برای چه هدفی باز نماییم؟ خواندن و یا نوشتن. برای روشن‌تر شده موضوع به مثال‌های زیر توجه کنید:

Open "C:\ VB\Data.txt" For Input As #1

Open "D:\Test.txt" For Output As #2

Open "E:\Programs\Phone.txt" For Append As #3

۱. **Input:** این حالت، فایل متنی مورد نظر را برای خواندن باز می‌کند. اگر فایل مورد نظر در مسیر خواسته شده موجود نباشد، برنامه با خطای خواهد شد.
۲. **Output:** این حالت، فایل متنی مورد نظر را برای نوشتن باز می‌کند. اگر فایل مورد نظر در مسیر خواسته شده موجود نباشد، ابتدا آن فایل را ایجاد و سپس آن را باز می‌کند. توجه کنید اگر فایل مورد نظر در مسیر خواسته شده وجود داشته باشد، پس از باز کردن، تمام محتويات آن را پاک می‌کند.
۳. **Append:** این حالت، فایل متنی مورد نظر را برای نوشتن باز می‌کند اما اطلاعات قبلی ذخیره شده در آن از بین نمی‌رود، بلکه اطلاعات جدید در ادامه‌ی اطلاعات قبلی نوشته می‌شوند. اگر فایل مورد نظر در مسیر خواسته شده موجود نباشد، یک پیغام خطای تولید خواهد شد.

- شماره‌ی فایل: عددی است که به یک فایل نسبت می‌دهیم و در برنامه برای ارتباط با فایل، از این شماره استفاده می‌کنیم و دیگر به نام فایل کاری نداریم.

۲.۳.۱۴. بستن فایل با دستور CLOSE

پس از استفاده از فایل باید آن را بیندید، تا اطلاعات داخل فایل صدمه نبیند و همچنین منابع سیستم به هدر نرود. با استفاده از دستور `Close #` می‌توانید فایل را که باز کردید، بیندید.

```
Open "C:\ VB\Data.txt" For Output As #1
```

```
Close #1
```

۳.۳.۱۴. نوشتن در فایل متنی با کمک دستور PRINT

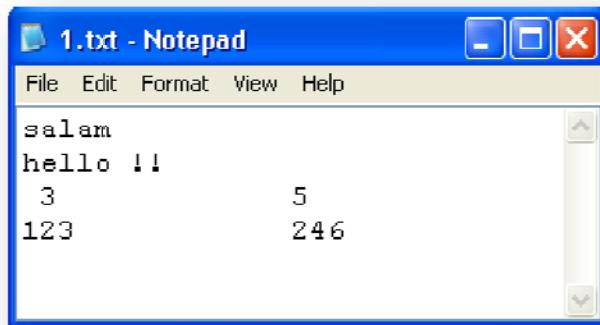
بعد از این‌که فایلی را با یکی از حالت‌های `Output` یا `Append` می‌توانید با استفاده از دستور `Print #`، داده‌های مورد نظر خود را درون آن بنویسید. نحوه‌ی استفاده از این دستور به شکل زیر است:

Print #، شماره‌ی فایل داده‌های مورد نظر

برای آزمایش دستورهای فوق، کافی است که قطعه برنامه زیر را در رخداد کلیک یک دکمه فرمان بنویسید. سپس برنامه را اجرا و بر روی دکمه فرمان مورد نظر کلیک نمایید. در مسیر داده شده برای ایجاد فایل، آن را به کمک Notepad باز نموده و اطلاعات آن را مشاهده نمایید.

```
Dim Temp As String
Dim Num As Integer
Open "C:\1.txt" For Output As #1
Print #1, "salam"
Temp = "hello !"
Print #1, Temp
Print #1, 1+2, 2+3
Num = 123
Print #1, Num, Num*2
Close #1
```

تصویر صفحه‌ی بعد، فایل ایجاد شده را در Notepad نشان می‌دهد.



دستور Print#، دقیقا مشابه دستور Print در بیسیک می‌باشد اما با این تفاوت که دستور Print در بیسیک داده‌ها را بر روی صفحه نمایش چاپ می‌کرد، اما دستور Print# داده‌ها را (با همان ساختار

دستور Print در فایل خروجی می‌نویسد. برای مثال اگر بین دو عدد داده یک علامت ویرگول قرار بگیرد، در فایل خروجی بین آن دو فاصله می‌افتد.

۴.۳.۱۴. خواندن از فایل متنی با دستور Input

بعد از نوشتن داده‌ها در فایل، طبیعتاً این انتظار می‌رود که بتوان آن‌ها را بازیابی نمود. برای این منظور فایل را با حالت Input باز کنید. دستور خواندن از یک فایل متنی Input# می‌باشد. دستور Input# نیز بسیار شبیه به دستور Input در بیسیک می‌باشد، فقط باید شماره فایل مورد نظر را به آن بدھید. نحوه‌ی استفاده از این دستور به شکل زیر است:

Input # نام متغیرهای مورد نظر ، **شماره‌ی فایل**

به عنوان مثال به عبارت‌های زیر دقت کنید:

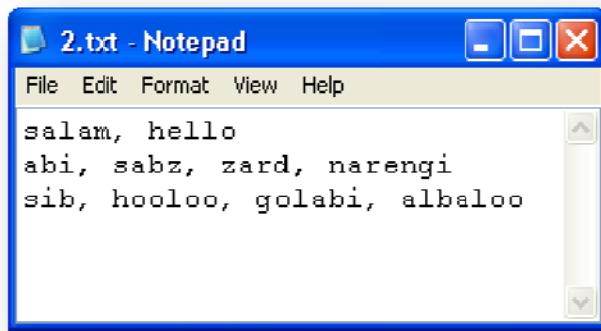
Input #1, A
Input #1, A, B

نکته :

هر متغیری که در جلوی دستور Input# قرار گرفته است، فقط یک خط از فایل متنی را می‌خواند؛ ولی به این نکته توجه داشته باشید که اگر در آن خط از فایل، علامت ویرگول وجود داشته باشد، دستور Input# فقط یکی از داده‌های مابین ویرگول‌ها را خوانده و برای خواندن بقیه‌ی داده‌ها، باید بار دیگر از این دستور استفاده نمایید.

اگر به خاطر آورید در بیسیک وقتی دستور Input a,b,c را می‌نوشتیم باید سه عدد به آن می‌دادیم و گرنۀ باعث تولید خطا می‌شد. اینجا نیز اگر نوع قرار گرفتن داده‌ها در فایل را بدانیم، خیلی راحت‌تر از دستور Input# استفاده خواهیم کرد.

به عنوان مثال، فایل متنی زیر را که در NotePad نمایش داده شده است، در نظر بگیرید:



این سه مثال تمام داده‌های موجود در فایل C:\2.txt را می‌خوانند.

نمونه‌ی ۱:

```
Open "C:\2.txt" For Input As #1
Input #1, A, B
Input #1, C, D, E, F
Input #1, G, H, I, J
Close #1
```

نمونه‌ی ۲:

```
Open "C:\2.txt" For Input As #1
Input #1, A
Input #1, B
Input #1, C
Input #1, D
Input #1, E
Input #1, F
Input #1, G
Input #1, H
Input #1, I
Input #1, J
Close #1
```


نمونه‌ی ۳:

```
Open "C:\2.txt" For Input As #1
While Not Eof(1)
    Input #1, A
    Wend
Close #1
```

نکته: بررسی رسیدن به انتهای فایل با کمک تابع EOF 

وقتی فایلی را برای خواندن باز می‌کنید، اشاره‌گری نیز ایجاد می‌شود که موقعیت داده‌ی خوانده شده از فایل را در خود ذخیره می‌نماید. با هر بار استفاده از دستورات Input# و Input# موقعیت Line Input# اشاره‌گر تغییر خواهد کرد. وقت کنید اگر پس از رسیدن اشاره‌گر به انتهای یک فایل، بار دیگر سعی در خواندن داده‌هایی از آن فایل داشته باشید، از آن جایی که اساساً دیگر داده‌ای وجود ندارد، اجرای برنامه با خطأ مواجه خواهد شد. بنابراین لازم است قبل از خواندن داده از فایل مطمئن باشید که اشاره‌گر به انتهای فایل نرسیده باشد. این کار به کمک تابع Eof قابل انجام است. در واقع تابع ([شماره فایل](#)) Eof شماره‌ی یک فایل را به عنوان ورودی گرفته و اگر اشاره‌گر فایل به انتهای آن فایل رسیده بود، مقدار True و در غیر این صورت مقدار False برمی‌گرداند. برای بررسی این موضوع کافی است از یک ساختار شرطی If استفاده نمایید:

خواندن از فایل مجاز است Else خواندن از فایل مجاز نیست Then ([شماره فایل](#))

۵.۳.۱۴. خواندن از فایل به کمک دستور **Line Input#**

دستور Line Input# یک خط کامل را از فایل می‌خواند، چه در آن خط ویرگول باشد و یا نباشد. بنابراین برنامه‌ی زیر نیز تمام داده‌های فایل C:\2.txt را می‌خواند، با این تفاوت که محتویات هر خط را در یک متغیر نگه می‌دارد:

```
Open "C:\2.txt" For Input As #1
Line Input #1, A
Line Input #1, B
Line Input #1, C
Close #1
```

۴.۱۴. آشنایی بیشتر با فایل‌ها

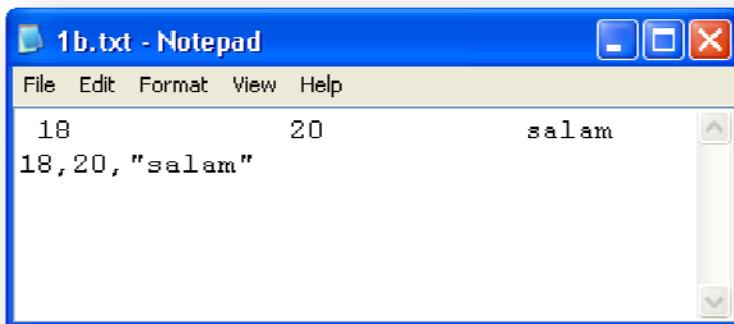
تا اینجا با مقدمات اصلی کار با فایل‌ها آشنا شده‌اید. در این قسمت به بعض نکات تکمیلی رابطه با فایل‌ها خواهیم پرداخت.

۱.۴.۱۴. نوشتن در فایل متى با دستور Write#

علاوه بر دستور Print#, دستور Write# نیز برای نوشتن در فایل وجود دارد. با توجه به مثال زیر، آیا می‌توانید تفاوت دو دستور فوق را حدس بزنید؟

```
Open "C:\1b.txt" For Output As #1
A = 18
B = 20
C ="salam"
Print #1, A, B, C
Write #1, A, B, C
Close #1
```

تصویر زیر خروجی برنامه‌ی فوق است:



دستور `#Write` علاوه بر این که داده‌ها را در فایل می‌نویسد، ویرگول‌ها را نیز بین آن‌ها، قرار می‌دهد و همچنین برای داده‌های رشته‌ای نیز در اطراف آن‌ها گیومه قرار می‌دهد. برای ایجاد فایل‌هایی با ساختار معین بهتر است از دستور `#Print` به جای دستور `#Write` استفاده کنید.

۲.۴.۱۴. شماره‌ی فایل و دستور FreeFile

اگر در برنامه‌ای قصد دارید چندین فایل را به طور همزمان، باز نگه‌دارید، باید برای هر کدام شماره‌ای یکتا در نظر بگیرید. بدین منظور باید شماره‌ی فایل‌های فایل‌های باز را بدانید و این کار بسیار مشکلی است. طراحان ویژوال بیسیک برای ساده کردن کار تابعی به نام `FreeFile` را ایجاد کرده‌اند. این تابع اولین شماره‌ی فایل آزاد را برمی‌گرداند. شکل استفاده از این تابع به صورت زیر است :

```
n = FreeFile
```

```
Open "C:\Test.txt" For Input As #n
```

در برنامه بالا ابتدا یک شماره فایل آزاد (اختصاص داده نشده به فایلی دیگر) در متغیر `n` ذخیره می‌شود. از آن پس به جای استفاده از مقدار شماره‌ی فایل، از این متغیر (که شماره فایل در آن ذخیره شده است) استفاده می‌کنید.

برای مثال جهت خواندن از فایل فوق کافی است بنویسید:

```
Line Input #n, A
```

۳.۴.۱۴. تابع LOF

این تابع شماره‌ی یک فایل را گرفته و حجم آن را بر حسب بایت برمی‌گرداند.

```
Dim Size As Long
```

```
Size = Lof(فایل)
```

Kill ۴.۴.۱۴. دستور

این دستور نام یک فایل را می‌گیرد و آن را حذف می‌کند.

Kill "C:\1b.txt"

۵.۱۴. کاراکترهای مهم در فایل‌های متنی

در فایل متنی، برای مشخص کردن انتهای خط و انتهای فایل، کاراکترهایی وجود دارد. برای مثال کد اسکی کاراکتر انتهای خط ۱۳ می‌باشد. دستور Line Input# یک خط از فایل متنی را می‌خواند و لی کاراکتر انتهای خط را نمی‌خواند. اگر می‌خواهید برنامه‌ای بنویسید که تمام خطوط یک فایل متنی را بخواند و در یک جعبه‌ی متن نمایش دهد باید کاراکتر انتهای خط را پس از خواندن هر خط به انتهای آن اضافه کنید. در ویژوال بیسیک برای اعداد و مقادیری که زیاد استفاده می‌شوند، ثابت‌هایی تعریف شده است، برای مثال vbNewLine ثابتی است که به جای (13) CHR\$(13) تعریف شده است.

۶.۱۴. تمرین

۱. دفترچه یادداشت:

- برنامه‌ای بنویسید که محتویات یک جعبه‌متن (TextBox) را درون یک فایل ذخیره کند.
- برنامه‌ای بنویسید که محتویات یک فایل متنی را بخواند و در یک جعبه‌متن نمایش دهد.

اکنون می‌توانید با کمک این دو برنامه، یک برنامه‌ی Notepad بنویسید.

احتمال دارد این دو برنامه در آینده به کارتان آید، بنابراین هر دو را ذخیره نمایید.

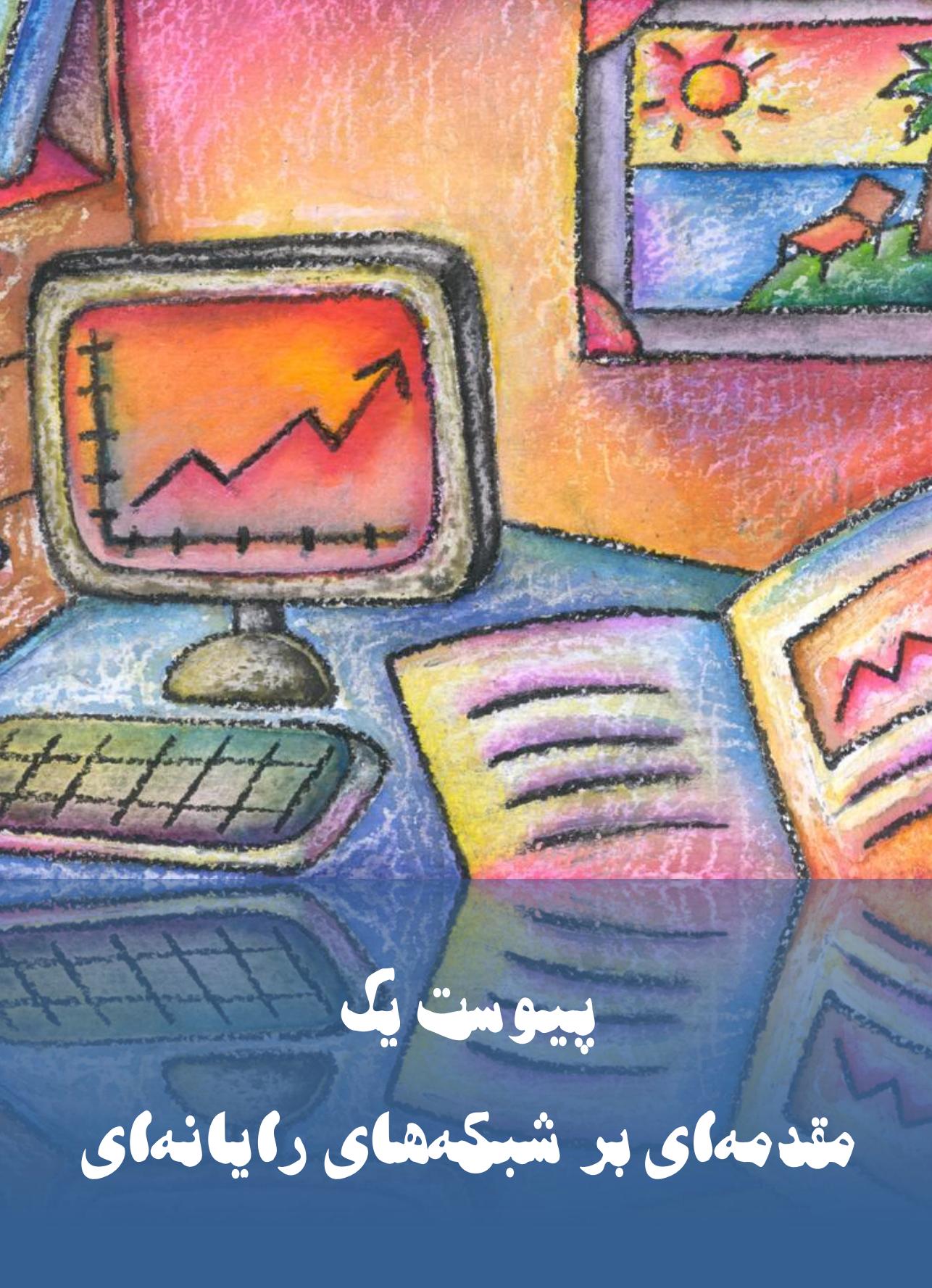
نکته:



اگر مقدار خصوصیت MultiLine مربوط به جعبه‌متن برابر True باشد می‌توان متون چند خطی در جعبه‌متن نوشت. خصوصیت ScrollBars نیز می‌تواند برای برنامه‌تان مفید باشد. پس از این که مقدار خصوصیت MultiLine را برابر True قرار دادید، خصوصیت ScrollBars را نیز بررسی کنید.

مقدمه‌ای بر شبکه‌های رایانه‌ای

پیوست یک



۱.۱۵. آشنایی

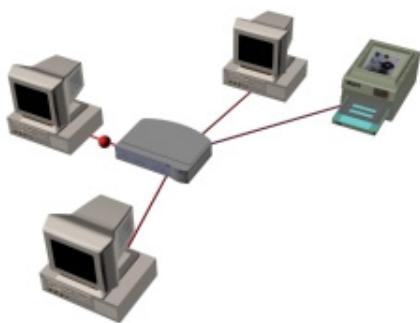
مفاهیم و برنامه‌نویسی شبکه یکی از جنبه‌های مطرح علم رایانه در دنیای کنونی است. امروزه این شاخه از دنیای برنامه‌نویسی، به سبب رشد و گسترش شبکه‌های محلی و در پی آن شبکه‌ی جهانی اینترنت، به اهمیت و جایگاه والایی رسیده است؛ تا آنجا که هم‌اکنون در دنیای فناوری اطلاعات و ارتباطات (ICT)، برنامه‌نویسی وب و به طور وسیع‌تر برنامه‌نویسی شبکه، تبدیل به یکی از مشاغل اصلی و کلیدی این حوزه شده است.

از آنجا که برای رسیدن به هر جایگاه با ارزشی، صبر، پشتکار و همت بلند لازم است، پس هیچ عجله نکنید. برای دستیابی به مقدمات برنامه‌نویسی شبکه در محیط ویژوال بیسیک، لازم است در ابتدا با مفاهیم پایه‌ی شبکه‌های رایانه‌ای آشنا شوید.

۱.۱۵. مفاهیم پایه‌ی شبکه‌های رایانه‌ای

متخصصان علم شبکه، شبکه را چنین تعریف کرده‌اند: شبکه‌ی رایانه‌ای، مجموعه‌ای از رایانه‌ها و سایر تجهیزات است که به یکدیگر متصل شده و می‌توانند با یکدیگر ارتباط برقرار کنند و همچنین منابع و اطلاعات خود را به اشتراک بگذارند.

تعریف دقیق‌تر و متداول‌تر شبکه در بین متخصصان عبارت است از: مجموعه‌ای از رایانه‌های مستقل که در ارتباط با یکدیگر، خدمات متنوعی را ارائه می‌دهند؛ که این خدمات شامل: رد و بدل کردن و انتقال اطلاعات، فراهم کردن محیطی برای ذخیره‌ی حجم بالای از اطلاعات، توزیع منابع موجود مانند چاپگر و در نهایت افزایش قدرت محاسبات و توان پردازشی تک‌تک رایانه‌ها است.



یک شبکه‌ی کوچک

یکی از دانشمندان قدیمی دنیای شبکه به نام Bob Metcalfe، از اولین کسانی بود که اعتقاد داشت ارزش یک شبکه، خیلی بیش‌تر از ارزش مجموعه رایانه‌های آن است. او که مبدع فناوری Ethernet و از مؤسسان شرکت 3Com است، درباره‌ی شبکه‌های رایانه‌ای چنین می‌گوید:

«هر رایانه‌ی که به یک شبکه می‌بینند، هم از شبکه به عنوان یک منبع استفاده می‌کند و هم خودش به عنوان یک منبع به شبکه اضافه می‌شود؛ ولذا قدرت یک شبکه، بسیار بیشتر از قدرت مجموع رایانه‌های آن است.»

در ادامه به بیان خصوصیات و مفاهیم پایه‌ی شبکه‌های رایانه‌ای می‌پردازیم.

۱۵. ۳. مزایای شبکه

برپاسازی یک شبکه‌ی رایانه‌ای دارای مزایای مهم زیر است:

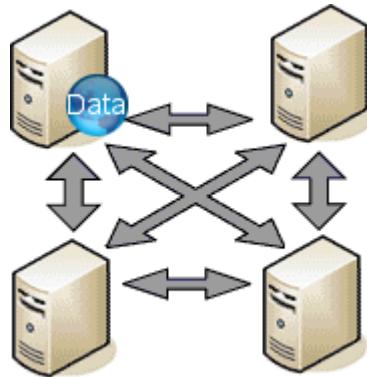
- **اشتراک اطلاعات:** امکان اشتراک اطلاعات و داده‌ها با سرعت مطلوب و هزینه‌ی پایین، از مهم‌ترین مزایای یک شبکه رایانه‌ای محسوب می‌شود.
- **اشتراک سخت‌افزار و نرم‌افزار:** قبل از مطرح شدن شبکه، کاربران از چاپگرهای سایر دستگاه‌های جانبی اختصاصی استفاده می‌کردند. رویکرد فوق، باعث افزایش هزینه‌ها، به‌خصوص در سازمان‌های بزرگ شده بود. با پیدایش و گسترش شبکه‌های رایانه‌ای، امکان استفاده از منابع سخت‌افزاری و نرم‌افزاری مشترک به صورت همزمان و توسط کاربران متعدد، فراهم گشت که باعث کاهش بسیاری از هزینه‌ها شد.
- **مدیریت و حمایت مرکزی:** برپا سازی یک شبکه، باعث ساده شدن مدیریت و عملیات مربوط به پشتیبانی می‌گردد. بدین ترتیب، مدیر شبکه از یک محل، قادر به انجام عملیات و وظایف مدیریتی بر روی هر یک از رایانه‌های موجود در شبکه، خواهد بود.

۴.۱۵ انواع شبکه

با توجه به نحوه‌ی دستیابی رایانه‌ها به اطلاعات، شبکه‌ها را به دو گروه عمده Peer-To-Peer و Client/Server تقسیم می‌کنند:

Peer-To-Peer (ناظیر به ناظیر): در شبکه‌های ناظیر به ناظیر، خدمات‌دهنده‌ی

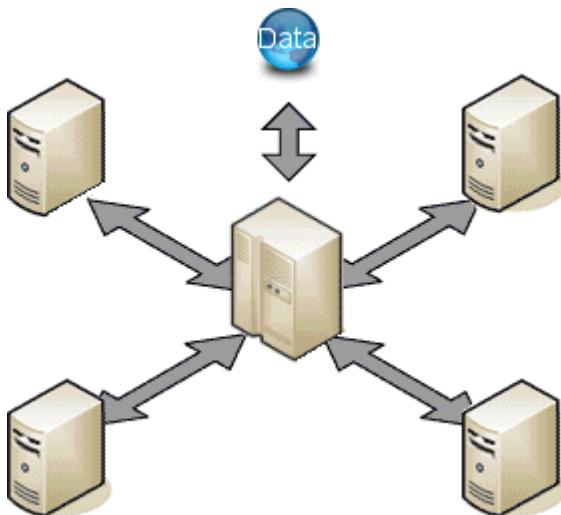
سرвис‌دهنده‌ی همان Server اختصاصی وجود نداشته و تمام رایانه‌ها معادل و همتراز می‌باشند. هر رایانه در سطح شبکه، هم به عنوان خدمات‌گیرنده (سرвис‌گیرنده‌ی یا مشتری اطلاعات) و هم به عنوان خدمات‌دهنده، ایفای وظیفه نموده و امنیت را به صورت محلی و بر روی هر رایانه تامین می‌کند؛ یعنی هر کاربر خود مشخص می‌کند که چه داده‌هایی از رایانه را می‌خواهد به اشتراک بگذارد. شبکه‌های ناظیر به ناظیر، ایستگاه کاری Workgroup، نیز نامیده می‌شوند. واژه Workgroup، نشان‌دهنده‌ی یک گروه کوچک (معمولًاً ده یا کمتر) از رایانه‌های مرتبط با یکدیگر است.



Client/Server (خدمات‌دهنده / خدمات‌گیرنده): به موازات رشد شبکه و افزایش تعداد

کاربران و منابع موجود، یک شبکه‌ی ناظیر به ناظیر، قادر به پاسخگویی حجم بالای تقاضاهای برای منابع اشتراکی نخواهد بود. به منظور بالابردن کارایی و ارائه‌ی خدمات مورد نیاز، شبکه‌ها می‌بایست از خدمات‌دهنگان اختصاصی، استفاده نمایند. یک خدمات‌دهنده‌ی اختصاصی، صرفاً به عنوان یک خدمات‌دهنده در شبکه، ایفای وظیفه می‌نماید (و نه به

عنوان یک خدمات‌گیرنده). شبکه‌های خدمات‌دهنده / خدمات‌گیرنده، به عنوان الگویی استاندارد برای بروزرسانی شبکه‌ها، مطرح شده‌اند. به موازات رشد شبکه (تعداد رایانه‌های متصل شده، فاصله‌ی فیزیکی و ترافیکی موجود) می‌توان تعداد خدمات‌دهنگان در شبکه را افزایش داد. با توزیع مناسب فعالیت‌های شبکه بین چندین خدمات‌دهنده، کارایی شبکه به طور محسوسی، افزایش خواهد یافت.



۱۵. ۵. آشنایی با بعضی اصطلاحات دنیای شبکه

- **پروتکل:** در لغت به معنی پیمان، معاہده و یا قرارداد است و در دنیای شبکه، عبارتست از قراردادی رایانه طبق آن با یکدیگر ارتباط برقرار کرده و در چارچوب قوانین آن به تبادل اطلاعات می‌پردازند. (به بیان ساده‌تر، پروتکل زبان مشترک بین رایانه‌های موجود در یک شبکه است)
- **پروتکل TCP/IP:** یک پروتکل جامع در اینترنت بوده و تمام کامپووترهایی که با اینترنت کار می‌کنند، از آن استفاده می‌کنند.
- **آدرس IP (IP Address):** در اینترنت هر رایانه دارای یک آدرس IP است. هر IP متشکل از چهار عدد بوده که با نقطه از هم جدا می‌شوند (برای مثال

آدرس است که هر رایانه در شبکه‌ها تعریف می‌شوند. (نظیر آدرس و شماره پلاک خانه‌ها در سطح شهر)

- **درگاه (Port):** نقش پل ارتباطی بین دو نقطه را ایفا می‌کند. درگاه و یا پورت معمولاً یک عدد چهار رقمی است و هر دو طرف ارتباط بایستی از یک درگاه مشترک (یک شماره‌ی واحد) استفاده کنند. به کمک درگاه‌های مختلف است که برنامه‌های مختلف در یک رایانه می‌توانند به روبدل کردن اطلاعات در شبکه بپردازند.
- **سوکت (Socket):** به بیان ساده، سوکت ارتباط شبکه با برنامه را باز می‌کند و اجازه می‌دهد تا برنامه، اطلاعات مورد نظر را از روی شبکه خوانده و یا بر روی آن بنویسید. در دنیای برنامه‌نویسی شبکه هیچ ارتباطی بدون سوکت برقرار نمی‌شود.

۱۵. چگونه نام یک رایانه را در شبکه پیدا کنیم ؟

به منظور اتصال به یک رایانه دیگر در شبکه، شما یا باید آدرس IP آن و یا اسم محلی آن را در شبکه پیدا کنید. معمولاً به خاطر سپردن اسم یک رایانه در سطح شبکه، راحت‌تر از به خاطر سپردن آدرس IP آن است.



با دنبال کردن مراحل زیر، نام رایانه‌ی خود را بیابید.

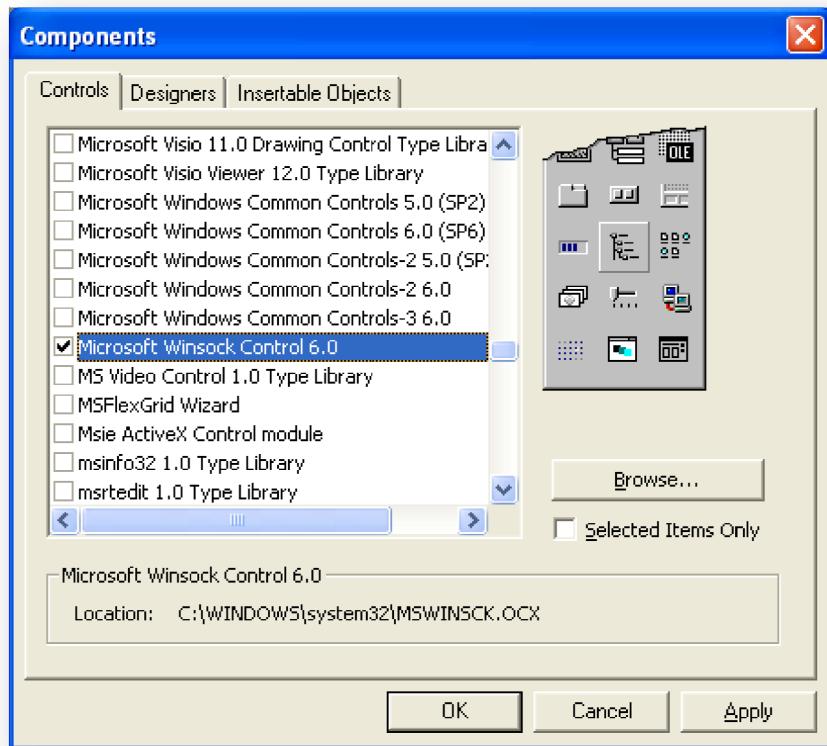
- بر روی دکمه Start کلیک کنید.
- وارد Control Panel شوید.
- بروزی نماد System دوکلیک کنید.
- بر روی بخش Computer Name کلیک کنید.
- نام رایانه در قسمت Full Computer Name نوشته شده است.

نکته :

برای به دست آوردن آدرس IP رایانه خود، باید بعد از وصل شدن به شبکه، در خط فرمان یا همان Command، دستور IPCONFIG را وارد کنید. با این کار، IP رایانه در قسمت IP Address IP Address نمایش داده می‌شود.

۱۵.۷. امکانات ویژوال بیسیک برای برنامه‌نویسی شبکه

کنترلی در ویژوال بیسیک به نام WinSock وجود دارد که به کمک آن می‌توان به رایانه‌های دیگر در شبکه متصل شده و از آن‌ها اطلاعات دریافت کرده و یا به آن‌ها اطلاعات فرستاد. برای استفاده از این کنترل ابتدا باید در ویژوال بیسیک، از منوی Project گزینه Components را انتخاب کنید. با این کار پنجره‌ای مطابق شکل ظاهر می‌شود.



پس از انتخاب گزینه Microsoft Winsock Control 6.0 (مطابق با شکل فوق) و کلیک کردن بر دکمه‌های **Apply** و **Close**، کنترل WinSock را در جعبه ابزار محیط ویژوال بیسیک مشاهده خواهید کرد. دقت کنید این کنترل در زمان اجرا بر روی فرم قابل رویت نیست.

به کمک توابع این کنترل نظیر `Connect` ، `SendData` و رخدادهای آن `ConnectionRequest` و `DataArrival` می‌توان به رویداد کردن اطلاعات در شبکه پرداخت. از آن جایی که مدل برنامه‌نویسی به کمک این کنترل از نوع خدمات دهنده / خدمات گیرنده (Client / Server) می‌باشد، قطعه کد هر دو برنامه را در ادامه بررسی خواهیم کرد. لازم به ذکر است که در این قطعه کدها فرض شده است که یک کنترل Winsock1 به اسم `Winsock1` در فرم موجود است. همچنین آدرس IP خدمات دهنده برابر `192.168.1.1` فرض شده و اطلاعات برروی درگاه `1001` ارسال می‌شوند. پروتکل قراردادی نیز `TCP/IP` می‌باشد.

کد مربوط به خدمات دهنده (Server)

```
Private Sub Form_Load()
    Winsock1.LocalPort = 1001
    Winsock1.Listen
End Sub
```

```
Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
    If Winsock1.State <> sckClosed Then Winsock1.Close
    Winsock1.Accept requestID
    Form1.Caption = requestID
End Sub
```

کد مربوط به خدمات گیرنده (Client)

```
Private Sub Form_Load()
    Winsock1.RemotePort = 1001
    Winsock1.RemoteHost = "192.168.1.1"
    Winsock1.Connect
End Sub
```

کد مربوط به دریافت و ارسال اطلاعات

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
    Dim s as String
    Winsock1.GetData s, vbString
    MsgBox s
End Sub

Private Sub Winsock1_Connect()
    Winsock1.SendData "SALAM!"
End Sub
```

Ctrl	Dec	Hex	Char	Code	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
^@	0	00		NUL	32	20	!	64	40	@	96	60	'
^A	1	01		SOH	33	21	..	65	41	A	97	61	a
^B	2	02		STX	34	22	#	66	42	B	98	62	b
^C	3	03		ETX	35	23	\$	67	43	C	99	63	c
^D	4	04		EOT	36	24	%	68	44	D	100	64	d
^E	5	05		ENQ	37	25	&	69	45	E	101	65	e
^F	6	06		ACK	38	26	,	70	46	F	102	66	f
^G	7	07		BEL	39	27	(71	47	G	103	67	g
^H	8	08		BS	40	28)	72	48	H	104	68	h
^I	9	09		HT	41	29	,	73	49	I	105	69	i
^J	10	0A		LF	42	2A	*	74	4A	J	106	6A	j
^K	11	0B		VT	43	2B	+	75	4B	K	107	6B	k
^L	12	0C		FF	44	2C	-	76	4C	L	108	6C	l
^M	13	0D		CR	45	2D	.	77	4D	M	109	6D	m
^N	14	0E		SO	46	2E	/	78	4E	N	110	6E	n
^O	15	0F		SI	47	2F	0	79	4F	O	111	6F	o
^P	16	10		DLE	48	30	1	80	50	P	112	70	p
^Q	17	11		DC1	49	31	2	81	51	Q	113	71	q
^R	18	12		DC2	50	32	3	82	52	R	114	72	r
^S	19	13		DC3	51	33	4	83	53	S	115	73	s
^T	20	14		DC4	52	34	5	84	54	T	116	74	t
^U	21	15		NAK	53	35	6	85	55	U	117	75	u
^V	22	16		SYN	54	36	7	86	56	V	118	76	v
^W	23	17		ETB	55	37	8	87	57	W	119	77	w
^X	24	18		CAN	56	38	9	88	58	X	120	78	x
^Y	25	19		EM	57	39	:	89	59	Y	121	79	y
^Z	26	1A		SUB	58	3A	:	90	5A	Z	122	7A	z
^[_	27	1B		ESC	59	3B	~	91	5B	[123	7B	{
^`	28	1C		FS	60	3C	<	92	5C	\	124	7C	
^]	29	1D		GS	61	3D	=	93	5D]	125	7D	}
^`	30	1E	▲	RS	62	3E	>	94	5E	~	126	7E	^*
^-	31	1F	▼	US	63	3F	?	95	5F	-	127	7F	Ø

* ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS).
The DEL code can be generated by the CTRL + BKSP key.

In the Name of God

STEP BY STEP PROGRAMMING WITH VISUAL BASIC

3RD GRADE

**Hamed Yaghoubi Shahir
Mohammad Siahatgar**

Sampad Publications

