توابع و دستورات در ویژوال بیسیک

دستور Print : این دستور برای نمایش اطلاعات روی صفحه نمایش به کار میرود. به طور کلی کاربرد این دستور را می توان در موارد زیر خلاصه کرد:

۱- نمایش رشته ها

۲- نمایش اعداد

در دستور پرینت از علامت کاما (,) و علامت سمیکالن(;) می توان استفاده کرد.

شکل کلی این دستور به صورت زیر است:

Print لیست داده ها شامل اعداد، متغیرها، عبارات و داده های رشته ای

در دستور فوق ، به جای کلمه کلیدی Print می توان از علامت سوال (؟) نیز استفاده کرد . چنانچه دستور Print بدون هیچ پارامتری به کار رود ، مکان نما را در خروجی از سطر جاری به سطر بعدی انتقال خواد داد.

از دستور Print می توان برای نمایش رشته ها و محتویات متغیر های رشته ای استفاده کرد که در این صورت رشته ها حتما باید در داخل علامت کوتیشن ("") قرار گیرند.

مثال) نمایش دو رشته و یک سطر خالی بین آنها:

```
Print "VB"
Print
Print "language"
```

خروجی:

```
VB

Language
```

علامت کاما و سمیکالن در دستور Print

استفاده از علامت کاما در دستور پرینت سبب جدا کردن مقادیر یا متغیرها از یکدیگر می شود که هر یک در ابتدای یک ناحیه، نمایش داده شوند. چنانچه بخواهیم مقادیر یا متغیر ها نزدیک تر به هم نمایش داده شوند در این

صورت می توان از علامت سمیکالن در دستور پرینت استفاده کرد. لازم به ذکر است که علامت کاما و سمیکالن توأما نیز می توانند در یک دستور پرینت جهت نمایش داده ها و متغیر ها به کار روند.

دستور exit

exit do

Exit for

Exit function

Exit property

Exit sub

سبب خروج از یک روال یا ساختا ر میشود.

دستور if

این دستور،شرطی راتست می کند و براساس نتیجه شرط (درستی یانادرستی)بخشی از برنامه را اجرا می کند.کاربرد های دستور if**به صورت زیر است:**

(۱)

if **شرط** then

مجموعه دستورات ۱

Else

مجموعه دستورات۲

End if

(۲)

if **شرط** Then

دستور ۱

دستور۲

.

.

.

Else

دستور ۱

دستور۲

.

.

.

End if

درکاربرداول چنانچه شرط درست باشد،مجموعه دستورات بعد از then**،وگرنه مجموعه دستورات بعد از** else **اجرا می شوند. درکاربرددوم نیز چنانچه شرط درست باشد،مجموعه دستورات بعد از** then**،وگرنه مجموعه دستورات بعد از** else **اجرا می شوند.**

تابع llf

این تابع همانند دستورif...else**عمل می کند با این تفاوت که میتوان از این تابع همواره در دستورات** if**با بدنه ی یک سطری استفاده کرد.تابع** llf **به صورت زیر به کار می رود:**

(دستور۲،دستور۱،شرط)llf

در این تابع چنانچه شرط درست باشد،دستور۱وگرنه دستور۲ اجرا می گردد.

دستور if **متداخل**

اگر بخواهیم از دستور if**برای تست شرط های متعددی استفاده کنیم باید آنها رابه طور تودرتو به کار ببریم .کاربرد** if **تودرتو نه تنهاموجب طولانی شدن برنامه می شود،بلکه ازخوانای برنامه نیز میکاهد.این ساختاربه صورت زیربه کارمی رود:**

if **شرط ۱** Then

دستورات ۱

else if**شرط ۲** Then

دستورات۲

.

.

.

else if n **شرط** Then

دستورات n

Else

دستورات else

End if

در این ساختار چنانچه شرط ۱درست باشد،دستورات ۱ اجرامی شوند وکنترل اجرای برنامه به بعد از end if**منتقل می شود وگرنه چنانچه شرط ۲درست باشد،دستورات۲اجرامی شوندودر غیر این صورت این روند ادامه می یابد.چنانچه هیچ یک از دستورات تا** n**درست نباشند،دستورات** else**اجرا می شوند.**

دستور Select Case

دستور If **یکی از دستورات شرطی** VB **می باشد. دیگر از دستورات شرطی، دستور** Select Case **است.**

جهت تغییر ترتیب اجرای برنامه، به جای استفاده از دستور If **می توان از دستور** Select case **استفاده کردتا بر اساس شرایط خاص، مجموعه ای از دستورات اجرا شوند.**

دستور Select Case **دارای شکل کلی زیر می باشد:**

Select Case **عبارت**

Case **عبارت ۱**

(مجموعه دستورات ۱)

Case (عبارت۲)

(مجموعه دستورات۲)

.

.

.

.

Case Else

مجموعه دستورات (n)

End Select

در ستور فوق ابتدا عبارت مقابل Select Case ارزیابی می شود و به محض برخورد با اولین Case که عبارت آن، شرایط عبارت مقابل Select Case را داشته باشد دستورات مربوط به آن Case اجرا خواهد شد، سپس کنترل اجرای برنامه به دستور بعد از End Select منتقل می گردد ور در صورتی که عبارتی پیدا نشود، دستورات بعد از Case Select اجرا خواهد شد و اجرای برنامه ادامه می باید.

تفاوت Select Case و IF در این است که در Select Case فقط یک عبارت مورد ارزیابی قرار گرفته، سپس مجموعه ای از دستورات اجرا می شود، اما در دستور IF چندین شرط مورد ارزیابی قرار میگیرد.

در مواردی که با تصمیم گیری های چند لایه ای سرو کار داریم، به کار بردن IF های بلوکی متعدد، برنامه را دچار ابهام و ناخوانایی می کند. برای رفع این مشکل از ساختار تصمیم گیری Select Case استفاده می شود که به سادگی شرایط تصمیم گیری چند لایه را می توان توسط آن پردازش نمود.

تابع inputbox

Inputbox(prompt,[title])

یک کادر محاوره ای را نمایش داده و منتظر دریافت متن از کاربر وکلیک روی دکمه اکی می شود. ارگومان prompt پیامی که باید نمایش داده شود تعیین می کند و tittle یک عنوان اختیاری برای نوار عنوان کادر محاوره های مشخص می کند .

دستور rem

Rem comments

به کمک این دستور میتوان توضیح هایی را به بر نامه اضافه کرد .هر چیزی که بعد از این دستور نوشته شود به وسیله ی ویژوال بیسیک صرف نظر می شود . به جای این دستور می توان از علامت اپستروف نیز استفاده کرد

دستور FOR … NEXT

دستور فوق جهت تکرار مجموعه ای از عملیات به تعداد مشخص، مورد استفاده قرار می گیرد. شکل کلی دستور FOR…NEXT

به صورت زیر می باشد:

For Variable = X to Y Step z

بلوک دستورالعمل ها {**دستورالعمل هایی که باید در داخل حلقه اجرا شوند.**

Next Variable

Variable **به عنوان متغیر شمارنده به کار میرود.که می تواند یک متغیر عددی صحیح یا اعشاری با دقت معمولی باشد.**

X **مقدار اولیه متغیر شمارنده را تعیین می کند که می تواند یک ثابت عددی، عبارت محاسباتی یا متغر عددی باشد.**

Y **مقدار نهایی متغیر شمارنده را تعیین می کند که می تواند یک ثابت عددی، عبارت محاسباتی یا متغر عددی باشد.**

Z **میزان افزایش متغر شمارنده را پس از تکرار حلقه در هر نوبت مشخص می کند. چنانچه جمله** Step z **در دستور** For **حذف شود، مفسر بیسیک میزان افزایش متغیر شمارنده** Z **را یک فرض خواهد نمود.** Z **می تواند ثابت عددی، عبارت محاسباتی یا متغیر عددی باشد.**

دستورات For **و**Next **همواره با هم و به صورت یک زوج به کار میروند و یکی بدون دیگری مفهومی ندارد. به این ترتیب عملیاتی که بین این دو دستور قرار می گیرند به تعداد دفعات معین تکرار می گردند.**

حلقه های متداخل (NESTED)

با استفاده از چندین دستور For وNext در داخل یکدیگر می توان حلقه های متداخل را ایجاد کرد که در این صورت به ازای هر بار افزایش متغیر شمارنده حلقه خارجی، حلقه داخلی به تعداد دفعات تعیین شده تکرار خواهد گردید.

For Variable = X to Y Step z

For variable J To H

دستورات داخل حلقه دوم

Next Variable J

Next Variable X

نکاتی درمورد حلقه forتودرتو

چنانچه حلقه های متداخل دارای نقطه پایان یکسانی باشند(دستوری بین next حلقه داخلی وخارجی نباشد)در این صورت یک دستور nextبه همراه کلیه متغیر های شمارنده برای تمامی حلقه ها کافی می باشد .مثل(next I,j).

دستور next حلقه داخلی با ید قبل از دستور next حلقه خارجی قرار گیرد .به عابرت دیگر حلقه نباید یکدیگررا قطع کنند.

دستور Do … Loop

این حلقه یکی از کاربردی ترین نوع حلقه های تکرار در برنامه نویسی است و معمولا به یکی از دو روش زیراستفاده می شود:

روش ۱ :

Do While **یا** Until **شرط**

بلوک حلقه {دستورالعمل های حلقه

Loop

روش ۲ :

Do

بلوک حلقه {دستورالعمل های حلقه

Loop (While یا Until شرط)

فرق بین دو روش ذکر شده در بررسی شرط حلقه می باشد. در روش اول ابتدا شرط بررسی شده و بعد دستورالعمل ها اجرا میگردند و ممکن است حالت هایی پیش آید که دستورالعمل های حلقه اصلا اجرا نگردند ولی در روش دوم شرط در انتهای حلقه کنترل می گردد و حداقل یک مرتبه دستورالعمل های حلقه اجرا می شنوند تا به عبارت While و یا Until برسد. در روش اول این عبارت مقابل Do و در روش دوم در مقابل عبارت Loop به کار می روند.

فرق عبارت While و Until این است که در While تا زمانی که شرط صحیح است دستورات بلوک حلقه، تکرار خواهند شد و به محض اینکه شرط نادرست گردید دیگر دستوالعمل ها اجرا نشده و کنترل اجرای برنامه به دستور بعد از Loop منتقل می گردد ولی در مورد Until بر عکس عمل می کند یعنی تا زمانی که شرط نادرست است دستورات بلوک حلقه، تکرار خواهند شد و به محض درستی شرط ، این دستوالعمل ها تکرار نشده و کنترل اجرای برنامه به اولین دستور بعد از Loop منتقل می گردد.

نکته: به کار بردن عبارت While یا Until به همراه شرط در Do … Loop اختیاری است.

مثال: برنامه زیر اعدادی را که مجذور آنها کوچکتر یا مساوی ۲۵ است به همراه مجزورشان نمایش می دهد.

```
Print "Example Of The Top-Test Form"
Print
Num% = 1
Do While Square% < 25
Square% = Num% ^ 2
Print "Number" ; Num% , "Square Of The Number Is " ; Square%
Num% = Num% + 1
Loop
```

خروجی:

```
Example Of The Top-Test Form
Number 1 square of number is 1
Number 2 square of number is 4
```

Number 3 square of number is 9

Number 4 square of number is 16

Number 5 square of number is 25

توابع ریاضی

همان گونه که گفته شد توابع ریاضی توابعی هستند که از قبل طراحی شده اند و برای انجام محاسبات روی عبارت های عددی مورد استفاده قرار می گیرند. نکته مهمی که باید در زمان استفاده از توابع دانست اینست که توابع همیشه یک مقدار را محاسبه کرده و باز می گردانند و این مقدار باز گردانده شده را می توان در داخل یک متغیر هم نوع ذخیره کرد و در برنامه مورد استفاده قرار داد. این توابع عبارتند از:

تابع ABS

این تابع مقدار قدر مطلق عبارت عددی یا عدد و یا متغیر عددی داخل پرانتز تابع را پس از محاسبه باز می گرداند.

مثال:

Print ABS (-3)

خروجی این مثال عدد ۳ بدون علامت است.

تابع SGN

این تابع علامت عدد یا متغیر عددی یا عبارت عددی داخل پرانتز تابع را پس از محاسبه باز می گرداند. شکل کلی استفاده از این تابع به صورت زیر است:

SGN (n)

n می تواند یک عدد و یا یک متغیر عددی و یا یک عبارت عددی باشد.

توجه داشته باشید که اگر n مثبت باشد مقدار ۱، اگر n منفی باشد مقدار -۱ و اگر n صفر باشد مقدار صفر باز گردانده می شود.

مثال:

Print SGN (-10)

خروجی مثال فوق عدد -۱ می باشد زیرا عددی که در تابع به کار رفته منفی است.

تابع SQR

این تابع جذر یا ریشه دوم عدد یا عبارت عددی یا متغیر عددی داخل پرانتز تابع را باز می گرداند و به شکل کلی زیر به کار می رود:

SQR (n)

n می تواند یک عدد یا یک متغیر عددی و یا یک عبارت عددی بزرگتر یا مساوی با صفر باشد.

مثال:

Print SQR (16)

خروجی مثال فوق عدد ۴ می باشد.

توجه داشته باشید که عدد داخل پرانتز SQR نباید منفی باشد زیرا اعداد منفی جذر ندارند. اگر این عدد منفی باشد از سوی مفسر VB پیغام خطا صادر می شود.

تابع FIX

این تابع قسمت صحیح عدد یا متغیر عددی و یا عبارت عددی داخل پرانتز تابع را باز می گرداند و به شکل کلی زیر به کار می رود:

FIX (n)

n می تواند یک عدد یا یک متغیر عددی و یا یک عبارت عددی باشد.

این تابع در واقع قسمت اعشاری عدد مورد نظر را حذف کرده و قسمت صحیح آن را باز می گرداند.

مثال:

Print FIX (12.49) , FIX (-18.72)

خروجی مثال فوق اعداد ۱۲و -۱۸ می باشد. زیرا تابع فوق فقط قسمت صحیح عدد را باز می گرداند.

تابع INT

این تابع بزرگترین عدد صحیح کوچکتر یا مساوی با عدد یا متغیر عددی یا عبارت عددی داخل پرانتز تابع را باز می گرداند و به شکل کلی زیر به کار می رود:

INT (n)

n می تواند یک عدد یا یک متغیر عددی و یا یک عبارت عددی باشد.

مثال:

```
Print INT (12.54) , INT (-99.4)
```

خروجی مثال فوق به ترتیب عدد ۱۲ و -۱۰۰ می باشد.

تابع CInt

این تابع عدد یا متغیر عددی یا عبارت عدد اعشاری داخل پرانتز تابع را گرد کرده و به یک عدد صحیح تبدیل می کند و به شکل کلی زیر به کار می رود:

```
CInt (n)
```

n می تواند یک عدد اعشاری یا یک متغیر اعشاری و یا یک عبارت عددی اعشاری باشد.

مثال:

```
Print CInt (45.67)
```

خروجی مثال فوق عدد ۴۶ است که گرد شده عدد ۴۵,۶۷ می باشد.

اگر n خارج از محدوده ۳۲۷۶۷ , -۳۲۷۶۸ باشد مفسر بیسیک پیغام خطای "Flow over" را نمایش می دهد.

تابع LOG

این تابع لگاریتم طبیعی عدد یا متغیر عددی یا عبارت عددی داخل پرانتز تابع را محاسبه می کند. عدد داخل پرانتز تابع باید بزرگ تر از صفر باشد و به شکل کلی زیر استفاده می شود:

```
LOG (n)
```

n می تواند یک عدد یا یک متغیر عددی و یا یک عبارت عددی بزرگتر از صفر باشد.

مثال:

```
Print LOG (2)
```

خروجی مثال فوق عدد ۰,۶۹۳۱۴۷۲ می باشد.

اگر n صفر یا عددی منفی باشد، پیغام خطا صادر خواهد شد.

تابع EXP

این تابع هر توانی از e **(پایه لگاریتم طبیعی که برابر است با ۲,۷۱۸۲۸۲) را محاسبه می کند و شکل کلی اسفاده از آن به صورت زیر است:**

EXP (n)

n **می تواند یک عدد و یا یک متغیر عددی و یا یک عبارت عددی کوچکتر یا مساوی ۸۸,۰۲۹۶۹ می تواند باشد.**

مثال:

X = 5

Print EXP (X – 1)

خروجی برنامه فوق عدد ۵۴,۵۹۸۱۵ می باشد.

تابع round

این تابع یک عبارت اعشاری راگرد می کندوبه صورت زیر به کار میرد

Round(exp1,n)

Exp1**مقدارعبارتی است که بایدگرد شودو**n**مشخص میکندکه این مقدار تا چند رقم بعد از آن باید گرد شود.اگررقم** n+1**بعداعشار بزرگتر یا مساوی۵ باشد،به رقم** n **امیک واحد اضافه شودوبه جای رقمهای که حذف می شوند،صفرقرارمیگیرد.دستورات زیررادر نظر بگیرید**

تابع rnd

این تابع یک عدد تصادفی بیین ۰ و ۱را بر میگرداندوبه صورت زیر به کار میرود:

Rnd(exp1)

Exp1 **مقدار عبارت عددی است که بر اساس ان بایدعددتصادفی تولید شود .**

توابع مثلثاتی

به آن دسته از توابع آماده ای گفته می شود که برای محاسبه نسبت های مثلثاتی به کار می روند.

توجه داشته باشید که در VB **زاویه بر حسب رادیان محاسبه می شود .**

تابع SIN

این تابع مقدار سینوس یک زاویه را باز می گرداند و به شکل کلی زیر به کار میرود:

SIN (**زاویه بر حسب رادیان**)

حاصل اجرای این تابع عددی اعشاری با دقت معمولی بین ۱, -۱ است

مثال:

```
Print SIN (1.5)
```

خروجی مثال فوق سینوس زاویه ۱,۵ رادیان است که برابر ۰,۹۹۷۴۹۵ می باشد.

تابع COS

این تابع مقدار کسینوس یک زاویه را باز می گرداند و به شکل کلی زیر به کار می رود:

COS (**زاویه بر حسب رادیان**)

مثال:

```
Print COS (3.1415)
```

خروجی برنامه فوق عدد -۱ می باشد.

تابع TAN

این تابع مقدار تان ژانت یک زاویه را باز می گرداند و به شکل کلی زیر به کار می رود:

TAN (**زاویه بر حسب رادیان**)

تابع ATN

این تابع مقدار آر کتانژانت نسبت دو ضلع مثلث قائم الزاویه را باز می گرداند. این تابع عکس تابع Tan عمل می کند و به شکل کلی زیر به کار میرود:

```
ATN (n)
```

n می تواند یک عدد یا یک متغیر عددی و یا یک عبارت عددی باشد.

توابع رشته ای

نویسه ها و رشته ها

رشته دنباله ای از نویسه هایی است که کنار هم قرار گرفته اند و مانند یک واحد عمل می کنند. یک رشته ممکن است شامل حروف، ارقام و انواع نویسه های خاص مانند $, \ , * , - , + و غیره باشد. رشته در ویژوال بیسیک از نوع داده String است.

در ویژوال بیسیک، رشته به صورت نویسه های متوالی که در یک زوج علامت کوتیشن قرار گرفته اند، نوشته می شود.

ویژوال بیسیک همچنین دارای متغیر هایی از نوع رشته ای است که می توانند هنگام اجرای برنامه، رشته های مختلفی را در خود جای دهند. پسوند نوع اعلان رشته، نماد $ (دلار) است.

الحاق رشته با & و +

با ترکیب چند رشته کوچکتر، می توان رشته های بزرگتر را ایجاد کرد. این عمل را می توان با استفاده از علامت جمع (+) یا امپرسند (&) انجام داد.

```
S1 = "Pro"
S2 = "gram"
S3 = s1 & s2
```

یا

```
S3 = s1 + s2
```

در عبارت فوق، دو رشته s1, s2 به یکدیگر الصاق شده و رشته جدید s3 را به وجود آورده اند که شامل Program است. اگر در هنگام الصاق رشته، دو عملوند موجود به صورت رشته باشند، استفاده از عملگر + و & فرقی با یکدیگر ندارند. با این وجود اگر عملگر + با عبارتی به کار رود که از نوع داده متفاوتی باشد، مساله ساز خواهد بود. به عنوان مثال، در عبارت:

```
S1 = "hello" + 22
```

ویژوال بیسیک ابتدا سعی می کند که رشته "hello" را به یک عدد تبدیل نماید و سپس با ۲۲ جمع کند؛ در حالی که رشته "hello" را نمی توان به عدد تبدیل کرد. بنابراین خطای عدم تطبیق (Mismatch Error) در اجرای برنامه رخ خواهد داد. بنابراین در الحاق رشته باید از عملگر & استفاده کرد.

نکته: همیشه از عملگر & برای الحاق رشته استفاده کنید.

مقایسه رشته ها

در کار کردن با رشته ها، مقایسه دو رشته از اهمیت خاصی برخوردار است. در ویژوال بیسیک، در مقایسهٔ رشته ها از تمام عملگر های مقایسه ای استفاده می کند.

برای مقایسه دو رشته از تابع StrComp استفاده می شود و شکی کلی آن به صورت زیر است:

Variant (**متغیریا صحیح**) = StrComp (String1, String2, [, Compare])

اگر String1 کوچکتر از String2 باشد، خروجی این تابع -۱ و اگر بزرگتر باشد، عدد ۱ است و در صورتی که دو رشته برابر باشند، خروجی صفر است. اگر هر یک از رشته ها Null باشند، خروجی تابع نیز Null خواهد بود.

تابع MID

ویژوال بیسیک برای کار کردن با نویسه های موجود در یک رشته دارای چندین ابزار تفاوت است. کار این تابع انتخاب یک زیر رشته از یک رشته می باشد و با استفاده از همین می توان یک رشته را معکوس کرد.

از تابع Len هم در این برنامه استفاده شده که این تابع اصول رشتهٔ دریافتی را بر میگرداند سپس با استفاده از تابع Mid در هر بار یک نویسه از رشته Phrase استخراج می کند.

کد برنامه به صورت زیر است:

```
Option Explicit
Private sub cmdReverse_Click()
Dim Phrase as string, position as integer
Txtoutput.text = ""
Phrase = txtInput.text
For position = Len (Phrase) To 1 Step -1
Txtoutput.text = txtOutput.text &_
Mid (Phrase, position, 1)
Next
End sub
```

توابع Right() و Left()

تابع Left ، سمت چپ ترین قسمت یک رشته را انتخاب و بر می گرداند. به عنوان مثال، اگر s1 و s2 متغیر های رشته ای باشند، عبارت های

```
S1 = "ABCDEF"
S2 = Left (s1, 4)
```

از سمت چپ رشتهٔ s1 چهار نویسهٔ ABCD را انتخاب و به s2 اختصاص می دهد. تابع Right، سمت راست ترین قسمت یک رشته را انتخاب و بر می گرداند. به عنوان مثال، در عبارت های

```
S1 = "ABCDEF"
S2 = Right (s1, 4)
```

از سمت راست رشتهٔ s1 چهار نویسهٔ CDEF انتخاب و به s2 اختصاص می یابد.

تابع len

```
Len(string/uarname)
```

یک مقدار long بر می گرداند که تعداد string نویسه های یا تعداد بایت های مورد نیاز برای ذخیره ی یک متغیر خاص(uarname)را مشخص می کند .

توابع Space() وString()

تابع String، رشته ای با تعداد نویسهٔ مشخص ایجاد می کند. تابع Space ، یک رشته که از فضاهای خالی تشکیل شده است، ایجاد می کند. برنامهٔ زیر، از این توابع استفاده می کند. هنگامی که برنامه اجرا شود، سه مورد روی فرم چاپ می شود. در خط ۳:

```
String (10 , "A")
```

رشته ای با ۱۰ حرف A ایجاد می شود. در صورتی که رشته بیش از یک نویسه داشته باشد، فقط نویسهٔ اول تکرار می شود.

در خط ۴:

```
String (5, 97)
```

رشته ای به طول ۵ نویسه با کد اسکی ۹۷ (حرف a) ایجاد می شود. در خط ۵:

Space (5) & String (5, "a")

رشته ای با ۵ فضای خالی و الصاق آن به رشته ای با ۵ حرف a **به وجود می آید.**

جایگزینی زیر رشته در یک رشته

Replace

واژه پرداز ها این توانایی را دارند که یک رشته را جستجو کرده و به جای آن، یک یا چند رشته جایگزین کنند. مثلا برای جایگزینی محتویات یک textbox **داخل یک** label **به صورت زیر عمل می کنیم.**

Label1.caption = Replace (Text1.text, " ", "..")

معکوس کردن یک رشته

StrReverse **همانطور که در برنامهٔ مربوط به تابع** mid **مشاهده کرده اید ، از تابع** mid **برای معکوس کردن رشته استفاده کردیم. ویژوال بیسیک دارای تابعی به نام** StrReverse **است که این کار را به خوبی انجام میدهد. مثلا می خواهید محتویات** Text1 **را در داخل** Text2 **معکوس کنید بدین صورت عمل می کنید.**

Text2.text = StrReverse (Text1.text)

تابع STR() **: این تابع، مقدار یک عبارت عددی را که به عنوان آرگومان دریافت می کند، به یک مقدار رشته ای تبدیل میکند.**

تابع CStr() **: آرگومان خود را به یک رشته تبدیل می کند.**

تابع VAL() **: این تابع برعکس** STR() **عمل می کند و یک رشتهٔ عددی را به عدد معادل آن تبدیل می کند.**

نکته: اگر درون رشته فضای خالی وجود داشته باشد، از آن صرف نظر می کند.

Print VAL ("12 45") «««««««« 1245

تبدیل نوع نویسه های رشته

تابع UCase() **تمام نویسه های آرگومان خود را به نویسه های بزرگ و تابع** LCase() **نویسه های آرگومان خود را به نویسه های کوچک تبدیل می کند.**