Visual Basic for Applications Reference

Visual Studio 6.0

Mid Function

See Also Example Specifics

Returns a **Variant** (**String**) containing a specified number of characters from a string.

Syntax

Mid(***string***, ***start***[, ***length***])

The **Mid** function syntax has these named arguments:

Part	Description
string	Required. String expression from which characters are returned. If ***string*** contains Null, **Null** is returned.
start	Required; Long. Character position in ***string*** at which the part to be taken begins. If ***start*** is greater than the number of characters in ***string***, **Mid** returns a zero-length string ("").
length	Optional; **Variant** (**Long**). Number of characters to return. If omitted or if there are fewer than ***length*** characters in the text (including the character at ***start***), all characters from the ***start*** position to the end of the string are returned.

Remarks

To determine the number of characters in ***string***, use the **Len** function.

Note Use the **MidB** function with byte data contained in a string, as in double-byte character set languages. Instead of specifying the number of characters, the arguments specify numbers of bytes. For sample code that uses **MidB**, see the second example in the example topic.

Visual Basic for Applications Reference

Mid Function Example

The first example uses the **Mid** function to return a specified number of characters from a string.

```
Dim MyString, FirstWord, LastWord, MidWords
MyString = "Mid Function Demo"   ' Create text string.
FirstWord = Mid(MyString, 1, 3)   ' Returns "Mid".
LastWord = Mid(MyString, 14, 4)   ' Returns "Demo".
MidWords = Mid(MyString, 5)   ' Returns "Function Demo".
```

The second example use **MidB** and a user-defined function (**MidMbcs**) to also return characters from string. The difference here is that the input string is ANSI and the length is in bytes.

```
Function MidMbcs(ByVal str as String, start, length)
   MidMbcs = StrConv(MidB(StrConv(str, vbFromUnicode), start, length), vbUnicode)
End Function

Dim MyString
MyString = "AbCdEfG"
' Where "A", "C", "E", and "G" are DBCS and "b", "d",
' and "f" are SBCS.
MyNewString = Mid(MyString, 3, 4)
' Returns ""CdEf"
MyNewString = MidB(MyString, 3, 4)
' Returns ""bC"
MyNewString = MidMbcs(MyString, 3, 4)
' Returns "bCd"
```

Visual Basic for Applications Reference

Visual Studio 6.0

Minute Function

See Also Example Specifics

Returns a **Variant** (**Integer**) specifying a whole number between 0 and 59, inclusive, representing the minute of the hour.

Syntax

Minute(*time***)**

The required *time* argument is any Variant, numeric expression, string expression, or any combination, that can represent a time. If *time* contains Null, **Null** is returned.

Visual Basic for Applications Reference

Minute Function Example

This example uses the **Minute** function to obtain the minute of the hour from a specified time. In the development environment, the time literal is displayed in short time format using the locale settings of your code.

```
Dim MyTime, MyMinute
MyTime = #4:35:17 PM#   ' Assign a time.
MyMinute = Minute(MyTime)   ' MyMinute contains 35.
```

Visual Basic for Applications Reference

Visual Studio 6.0

MIRR Function

See Also Example Specifics

Returns a Double specifying the modified internal rate of return for a series of periodic cash flows (payments and receipts).

Syntax

MIRR(*values*(), *finance_rate*, *reinvest_rate*)

The **MIRR** function has these named arguments:

Part	Description
***values*()**	Required. Array of **Double** specifying cash flow values. The array must contain at least one negative value (a payment) and one positive value (a receipt).
finance_rate	Required. **Double** specifying interest rate paid as the cost of financing.
reinvest_rate	Required. **Double** specifying interest rate received on gains from cash reinvestment.

Remarks

The modified internal rate of return is the internal rate of return when payments and receipts are financed at different rates. The **MIRR** function takes into account both the cost of the investment (***finance_rate***) and the interest rate received on reinvestment of cash (***reinvest_rate***).

The ***finance_rate*** and ***reinvest_rate*** arguments are percentages expressed as decimal values. For example, 12 percent is expressed as 0.12.

The **MIRR** function uses the order of values within the array to interpret the order of payments and receipts. Be sure to enter your payment and receipt values in the correct sequence.

Visual Basic for Applications Reference

MIRR Function Example

This example uses the **MIRR** function to return the modified internal rate of return for a series of cash flows contained in the array `Values()`. `LoanAPR` represents the financing interest, and `InvAPR` represents the interest rate received on reinvestment.

```
Dim LoanAPR, InvAPR, Fmt, RetRate, Msg
Static Values(5) As Double   ' Set up array.
LoanAPR = .1   ' Loan rate.
InvAPR = .12   ' Reinvestment rate.
Fmt = "#0.00"   ' Define money format.
Values(0) = -70000   ' Business start-up costs.
' Positive cash flows reflecting income for four successive years.
Values(1) = 22000 : Values(2) = 25000
Values(3) = 28000 : Values(4) = 31000
RetRate = MIRR(Values(), LoanAPR, InvAPR)   ' Calculate internal rate.
Msg = "The modified internal rate of return for these five cash flows is"
Msg = Msg & Format(Abs(RetRate) * 100, Fmt) & "%."
MsgBox Msg   ' Display internal return
      ' rate.
```

Visual Basic for Applications Reference

Visual Studio 6.0

Month Function

See Also Example Specifics

Returns a **Variant** (**Integer**) specifying a whole number between 1 and 12, inclusive, representing the month of the year.

Syntax

Month(*date***)**

The required *date* argument is any Variant, numeric expression, string expression, or any combination, that can represent a date. If *date* contains Null, **Null** is returned.

Note If the **Calendar** property setting is Gregorian, the returned integer represents the Gregorian day of the week for the date argument. If the calendar is Hijri, the returned integer represents the Hijri day of the week for the date argument. For Hijri dates, the argument number is any numeric expression that can represent a date and/or time from 1/1/100 (Gregorian Aug 2, 718) through 4/3/9666 (Gregorian Dec 31, 9999).

Visual Basic for Applications Reference

Month Function Example

This example uses the **Month** function to obtain the month from a specified date. In the development environment, the date literal is displayed in short date format using the locale settings of your code.

```
Dim MyDate, MyMonth
MyDate = #February 12, 1969#   ' Assign a date.
MyMonth = Month(MyDate)   ' MyMonth contains 2.
```

Visual Basic for Applications Reference

Visual Studio 6.0

MonthName Function

See Also Example Specifics

Description

Returns a string indicating the specified month.

Syntax

MonthName(*month*[**,** *abbreviate*]**)**

The **MonthName** function syntax has these parts:

Part	Description
month	Required. The numeric designation of the month. For example, January is 1, February is 2, and so on.
abbreviate	Optional. **Boolean** value that indicates if the month name is to be abbreviated. If omitted, the default is **False**, which means that the month name is not abbreviated.

Visual Basic for Applications Reference

Visual Studio 6.0

MsgBox Function

See Also Example Specifics

Displays a message in a dialog box, waits for the user to click a button, and returns an **Integer** indicating which button the user clicked.

Syntax

MsgBox(*prompt*[, *buttons*] [, *title*] [, *helpfile*, *context*])

The **MsgBox** function syntax has these named arguments:

Part	Description
prompt	Required. String expression displayed as the message in the dialog box. The maximum length of ***prompt*** is approximately 1024 characters, depending on the width of the characters used. If ***prompt*** consists of more than one line, you can separate the lines using a carriage return character (**Chr(**13**)**), a linefeed character (**Chr(**10**)**), or carriage return linefeed character combination (**Chr(**13**)** & **Chr(**10**)**) between each line.
buttons	Optional. Numeric expression that is the sum of values specifying the number and type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. If omitted, the default value for ***buttons*** is 0.
title	Optional. String expression displayed in the title bar of the dialog box. If you omit ***title***, the application name is placed in the title bar.
helpfile	Optional. String expression that identifies the Help file to use to provide context-sensitive Help for the dialog box. If ***helpfile*** is provided, ***context*** must also be provided.
context	Optional. Numeric expression that is the Help context number assigned to the appropriate Help topic by the Help author. If ***context*** is provided, ***helpfile*** must also be provided.

Settings

The ***buttons*** argument settings are:

Constant	Value	Description
vbOKOnly	0	Display **OK** button only.
vbOKCancel	1	Display **OK** and **Cancel** buttons.

vbAbortRetryIgnore	2	Display **Abort**, **Retry**, and **Ignore** buttons.
vbYesNoCancel	3	Display **Yes**, **No**, and **Cancel** buttons.
vbYesNo	4	Display **Yes** and **No** buttons.
vbRetryCancel	5	Display **Retry** and **Cancel** buttons.
vbCritical	16	Display **Critical Message** icon.
vbQuestion	32	Display **Warning Query** icon.
vbExclamation	48	Display **Warning Message** icon.
vbInformation	64	Display **Information Message** icon.
vbDefaultButton1	0	First button is default.
vbDefaultButton2	256	Second button is default.
vbDefaultButton3	512	Third button is default.
vbDefaultButton4	768	Fourth button is default.
vbApplicationModal	0	Application modal; the user must respond to the message box before continuing work in the current application.
vbSystemModal	4096	System modal; all applications are suspended until the user responds to the message box.
vbMsgBoxHelpButton	16384	Adds Help button to the message box
VbMsgBoxSetForeground	65536	Specifies the message box window as the foreground window
vbMsgBoxRight	524288	Text is right aligned
vbMsgBoxRtlReading	1048576	Specifies text should appear as right-to-left reading on Hebrew and Arabic systems

The first group of values (05) describes the number and type of buttons displayed in the dialog box; the second group (16, 32, 48, 64) describes the icon style; the third group (0, 256, 512) determines which button is the default; and the fourth group (0, 4096) determines the modality of the message box. When adding numbers to create a final value for the ***buttons*** argument, use only one number from each group.

Note These constants are specified by Visual Basic for Applications. As a result, the names can be used anywhere in your code in place of the actual values.

Return Values

Constant	Value	Description
vbOK	1	**OK**

vbCancel	2	**Cancel**
vbAbort	3	**Abort**
vbRetry	4	**Retry**
vbIgnore	5	**Ignore**
vbYes	6	**Yes**
vbNo	7	**No**

Remarks

When both ***helpfile*** and ***context*** are provided, the user can press F1 to view the Help topic corresponding to the **context**. Some host applications, for example, Microsoft Excel, also automatically add a **Help** button to the dialog box.

If the dialog box displays a **Cancel** button, pressing the ESC key has the same effect as clicking **Cancel**. If the dialog box contains a **Help** button, context-sensitive Help is provided for the dialog box. However, no value is returned until one of the other buttons is clicked.

Note To specify more than the first named argument, you must use **MsgBox** in an expression. To omit some positional arguments, you must include the corresponding comma delimiter.

Visual Basic for Applications Reference

MsgBox Function Example

This example uses the **MsgBox** function to display a critical-error message in a dialog box with Yes and No buttons. The No button is specified as the default response. The value returned by the **MsgBox** function depends on the button chosen by the user. This example assumes that DEMO.HLP is a Help file that contains a topic with a Help context number equal to 1000.

```
Dim Msg, Style, Title, Help, Ctxt, Response, MyString
Msg = "Do you want to continue ?"   ' Define message.
Style = vbYesNo + vbCritical + vbDefaultButton2   ' Define buttons.
Title = "MsgBox Demonstration"   ' Define title.
Help = "DEMO.HLP"   ' Define Help file.
Ctxt = 1000   ' Define topic
      ' context.
      ' Display message.
Response = MsgBox(Msg, Style, Title, Help, Ctxt)
If Response = vbYes Then   ' User chose Yes.
   MyString = "Yes"   ' Perform some action.
Else   ' User chose No.
   MyString = "No"   ' Perform some action.
End If
```

Visual Basic for Applications Reference

Visual Studio 6.0

Now Function

See Also Example Specifics

Returns a **Variant** (**Date**) specifying the current date and time according your computer's system date and time.

Syntax

Now

Visual Basic for Applications Reference

Now Function Example

This example uses the **Now** function to return the current system date and time.

```
Dim Today
Today = Now    ' Assign current system date and time.
```

Visual Basic for Applications Reference

Visual Studio 6.0

NPer Function

See Also Example Specifics

Returns a Double specifying the number of periods for an annuity based on periodic, fixed payments and a fixed interest rate.

Syntax

NPer(*rate*, *pmt*, *pv*[, *fv*[, *type*]])

The **NPer** function has these named arguments:

Part	Description
rate	Required. **Double** specifying interest rate per period. For example, if you get a car loan at an annual percentage rate (APR) of 10 percent and make monthly payments, the rate per period is 0.1/12, or 0.0083.
pmt	Required. **Double** specifying payment to be made each period. Payments usually contain principal and interest that doesn't change over the life of the annuity.
pv	Required. **Double** specifying present value, or value today, of a series of future payments or receipts. For example, when you borrow money to buy a car, the loan amount is the present value to the lender of the monthly car payments you will make.
fv	Optional. Variant specifying future value or cash balance you want after you've made the final payment. For example, the future value of a loan is $0 because that's its value after the final payment. However, if you want to save $50,000 over 18 years for your child's education, then $50,000 is the future value. If omitted, 0 is assumed.
type	Optional. **Variant** specifying when payments are due. Use 0 if payments are due at the end of the payment period, or use 1 if payments are due at the beginning of the period. If omitted, 0 is assumed.

Remarks

An annuity is a series of fixed cash payments made over a period of time. An annuity can be a loan (such as a home mortgage) or an investment (such as a monthly savings plan).

For all arguments, cash paid out (such as deposits to savings) is represented by negative numbers; cash received (such as dividend checks) is represented by positive numbers.

Visual Basic for Applications Reference

NPer Function Example

This example uses the **NPer** function to return the number of periods during which payments must be made to pay off a loan whose value is contained in `PVal`. Also provided are the interest percentage rate per period (`APR / 12`), the payment (`Payment`), the future value of the loan (`FVal`), and a number that indicates whether the payment is due at the beginning or end of the payment period (`PayType`).

```
Dim FVal, PVal, APR, Payment, PayType, TotPmts
Const ENDPERIOD = 0, BEGINPERIOD = 1   ' When payments are made.
FVal = 0   ' Usually 0 for a loan.
PVal = InputBox("How much do you want to borrow?")
APR = InputBox("What is the annual percentage rate of your loan?")
If APR > 1 Then APR = APR / 100   ' Ensure proper form.
Payment = InputBox("How much do you want to pay each month?")
PayType = MsgBox("Do you make payments at the end of month?", vbYesNo)
If PayType = vbNo Then PayType = BEGINPERIOD Else PayType = ENDPERIOD
TotPmts = NPer(APR / 12, -Payment, PVal, FVal, PayType)
If Int(TotPmts) <> TotPmts Then TotPmts = Int(TotPmts) + 1
MsgBox "It will take you " & TotPmts & " months to pay off your loan."
```

Visual Basic for Applications Reference

Visual Studio 6.0

NPV Function

See Also Example Specifics

Returns a Double specifying the net present value of an investment based on a series of periodic cash flows (payments and receipts) and a discount rate.

Syntax

NPV(*rate*, *values*())

The **NPV** function has these named arguments:

Part	Description
rate	Required. **Double** specifying discount rate over the length of the period, expressed as a decimal.
***values*()**	Required. Array of **Double** specifying cash flow values. The array must contain at least one negative value (a payment) and one positive value (a receipt).

Remarks

The net present value of an investment is the current value of a future series of payments and receipts.

The **NPV** function uses the order of values within the array to interpret the order of payments and receipts. Be sure to enter your payment and receipt values in the correct sequence.

The **NPV** investment begins one period before the date of the first cash flow value and ends with the last cash flow value in the array.

The net present value calculation is based on future cash flows. If your first cash flow occurs at the beginning of the first period, the first value must be added to the value returned by **NPV** and must not be included in the cash flow values of ***values*( )**.

The **NPV** function is similar to the **PV** function (present value) except that the **PV** function allows cash flows to begin either at the end or the beginning of a period. Unlike the variable **NPV** cash flow values, **PV** cash flows must be fixed throughout the investment.

Visual Basic for Applications Reference

NPV Function Example

This example uses the **NPV** function to return the net present value for a series of cash flows contained in the array `Values()`. `RetRate` represents the fixed internal rate of return.

```
Dim Fmt, Guess, RetRate, NetPVal, Msg
Static Values(5) As Double   ' Set up array.
Fmt = "###,##0.00"   ' Define money format.
Guess = .1   ' Guess starts at 10 percent.
RetRate = .0625   ' Set fixed internal rate.
Values(0) = -70000   ' Business start-up costs.
' Positive cash flows reflecting income for four successive years.
Values(1) = 22000 : Values(2) = 25000
Values(3) = 28000 : Values(4) = 31000
NetPVal = NPV(RetRate, Values())   ' Calculate net present value.
Msg = "The net present value of these cash flows is "
Msg = Msg & Format(NetPVal, Fmt) & "."
MsgBox Msg   ' Display net present value.
```

Visual Basic for Applications Reference

Visual Studio 6.0

Oct Function

See Also Example Specifics

Returns a **Variant** (**String**) representing the octal value of a number.

Syntax

Oct(*number*)

The required *number* argument is any valid numeric expression or string expression.

Remarks

If *number* is not already a whole number, it is rounded to the nearest whole number before being evaluated.

If *number* is	**Oct returns**
Null	**Null**
Empty	Zero (0)
Any other number	Up to 11 octal characters

You can represent octal numbers directly by preceding numbers in the proper range with &0. For example, &010 is the octal notation for decimal 8.

Visual Basic for Applications Reference

Oct Function Example

This example uses the **Oct** function to return the octal value of a number.

```
Dim MyOct
MyOct = Oct(4)    ' Returns 4.
MyOct = Oct(8)   ' Returns 10.
MyOct = Oct(459)   ' Returns 713.
```

Visual Basic for Applications Reference

Visual Studio 6.0

Partition Function

See Also Example Specifics

Returns a **Variant** (**String**) indicating where a number occurs within a calculated series of ranges.

Syntax

Partition(*number*, *start*, *stop*, *interval*)

The **Partition** function syntax has these named arguments:

Part	Description
number	Required. Whole number that you want to evaluate against the ranges.
start	Required. Whole number that is the start of the overall range of numbers. The number can't be less than 0.
stop	Required. Whole number that is the end of the overall range of numbers. The number can't be equal to or less than ***start***.

Remarks

The **Partition** function identifies the particular range in which ***number*** falls and returns a **Variant** (**String**) describing that range. The **Partition** function is most useful in queries. You can create a select query that shows how many orders fall within various ranges, for example, order values from 1 to 1000, 1001 to 2000, and so on.

The following table shows how the ranges are determined using three sets of ***start***, ***stop***, and ***interval*** parts. The First Range and Last Range columns show what **Partition** returns. The ranges are represented by *lowervalue*:*uppervalue*, where the low end (*lowervalue*) of the range is separated from the high end (*uppervalue*) of the range with a colon (**:**).

start	*stop*	*interval*	Before First	First Range	Last Range	After Last
0	99	5	" :-1"	" 0: 4"	" 95: 99"	" 100: "
20	199	10	" : 19"	" 20: 29"	" 190: 199"	" 200: "
100	1010	20	" : 99"	" 100: 119"	" 1000: 1010"	" 1011: "

In the table shown above, the third line shows the result when ***start*** and ***stop*** define a set of numbers that can't be evenly divided by ***interval***. The last range extends to ***stop*** (11 numbers) even though ***interval*** is 20.

If necessary, **Partition** returns a range with enough leading spaces so that there are the same number of characters to the left and right of the colon as there are characters in ***stop***, plus one. This ensures that if you use **Partition** with other numbers, the resulting text will be handled properly during any subsequent sort operation.

If ***interval*** is 1, the range is ***number:number***, regardless of the ***start*** and ***stop*** arguments. For example, if ***interval*** is 1, ***number*** is 100 and ***stop*** is 1000, **Partition** returns " 100: 100".

If any of the parts is Null, **Partition** returns a **Null**.

Visual Basic for Applications Reference

Partition Function Example

This example assumes you have an Orders table that contains a Freight field. It creates a select procedure that counts the number of orders for which freight cost falls into each of several ranges. The **Partition** function is used first to establish these ranges, then the SQL Count function counts the number of orders in each range. In this example, the arguments to the **Partition** function are *start* = 0, *stop* = 500, *interval* = 50. The first range would therefore be 0:49, and so on up to 500.

```
SELECT DISTINCTROW Partition([freight],0, 500, 50) AS Range,
Count(Orders.Freight) AS Count
FROM Orders
GROUP BY Partition([freight],0,500,50);
```

Visual Basic for Applications Reference

Visual Studio 6.0

Pmt Function

See Also Example Specifics

Returns a Double specifying the payment for an annuity based on periodic, fixed payments and a fixed interest rate.

Syntax

Pmt(*rate*, *nper*, *pv*[, *fv*[, *type*]])

The **Pmt** function has these named arguments:

Part	Description
rate	Required. **Double** specifying interest rate per period. For example, if you get a car loan at an annual percentage rate (APR) of 10 percent and make monthly payments, the rate per period is 0.1/12, or 0.0083.
nper	Required. Integer specifying total number of payment periods in the annuity. For example, if you make monthly payments on a four-year car loan, your loan has a total of 4 * 12 (or 48) payment periods.
pv	Required. **Double** specifying present value (or lump sum) that a series of payments to be paid in the future is worth now. For example, when you borrow money to buy a car, the loan amount is the present value to the lender of the monthly car payments you will make.
fv	Optional. Variant specifying future value or cash balance you want after you've made the final payment. For example, the future value of a loan is $0 because that's its value after the final payment. However, if you want to save $50,000 over 18 years for your child's education, then $50,000 is the future value. If omitted, 0 is assumed.
type	Optional. **Variant** specifying when payments are due. Use 0 if payments are due at the end of the payment period, or use 1 if payments are due at the beginning of the period. If omitted, 0 is assumed.

Remarks

An annuity is a series of fixed cash payments made over a period of time. An annuity can be a loan (such as a home mortgage) or an investment (such as a monthly savings plan).

The ***rate*** and ***nper*** arguments must be calculated using payment periods expressed in the same units. For example, if ***rate*** is calculated using months, ***nper*** must also be calculated using months.

For all arguments, cash paid out (such as deposits to savings) is represented by negative numbers; cash received (such as dividend checks) is represented by positive numbers.

Visual Basic for Applications Reference

Pmt Function Example

This example uses the **Pmt** function to return the monthly payment for a loan over a fixed period. Given are the interest percentage rate per period (`APR / 12`), the total number of payments (`TotPmts`), the present value or principal of the loan (`PVal`), the future value of the loan (`FVal`), and a number that indicates whether the payment is due at the beginning or end of the payment period (`PayType`).

```
Dim Fmt, FVal, PVal, APR, TotPmts, PayType, Payment
Const ENDPERIOD = 0, BEGINPERIOD = 1   ' When payments are made.
Fmt = "###,###,##0.00"   ' Define money format.
FVal = 0   ' Usually 0 for a loan.
PVal = InputBox("How much do you want to borrow?")
APR = InputBox("What is the annual percentage rate of your loan?")
If APR > 1 Then APR = APR / 100   ' Ensure proper form.
TotPmts = InputBox("How many monthly payments will you make?")
PayType = MsgBox("Do you make payments at the end of month?", vbYesNo)
If PayType = vbNo Then PayType = BEGINPERIOD Else PayType = ENDPERIOD
Payment = Pmt(APR / 12, TotPmts, -PVal, FVal, PayType)
MsgBox "Your payment will be " & Format(Payment, Fmt) & " per month."
```

Visual Basic for Applications Reference

Visual Studio 6.0

PPmt Function

See Also Example Specifics

Returns a Double specifying the principal payment for a given period of an annuity based on periodic, fixed payments and a fixed interest rate.

Syntax

PPmt(*rate*, *per*, *nper*, *pv*[, *fv*[, *type*]])

The **PPmt** function has these named arguments:

Part	Description
rate	Required. **Double** specifying interest rate per period. For example, if you get a car loan at an annual percentage rate (APR) of 10 percent and make monthly payments, the rate per period is 0.1/12, or 0.0083.
per	Required. Integer specifying payment period in the range 1 through ***nper***.
nper	Required. **Integer** specifying total number of payment periods in the annuity. For example, if you make monthly payments on a four-year car loan, your loan has a total of 4 * 12 (or 48) payment periods.
pv	Required. **Double** specifying present value, or value today, of a series of future payments or receipts. For example, when you borrow money to buy a car, the loan amount is the present value to the lender of the monthly car payments you will make.
fv	Optional. Variant specifying future value or cash balance you want after you've made the final payment. For example, the future value of a loan is $0 because that's its value after the final payment. However, if you want to save $50,000 over 18 years for your child's education, then $50,000 is the future value. If omitted, 0 is assumed.
type	Optional. **Variant** specifying when payments are due. Use 0 if payments are due at the end of the payment period, or use 1 if payments are due at the beginning of the period. If omitted, 0 is assumed.

Remarks

An annuity is a series of fixed cash payments made over a period of time. An annuity can be a loan (such as a home mortgage) or an investment (such as a monthly savings plan).

The ***rate*** and ***nper*** arguments must be calculated using payment periods expressed in the same units. For example, if ***rate*** is calculated using months, ***nper*** must also be calculated using months.

For all arguments, cash paid out (such as deposits to savings) is represented by negative numbers; cash received (such as dividend checks) is represented by positive numbers.

Visual Basic for Applications Reference

PPmt Function Example

This example uses the **PPmt** function to calculate how much of a payment for a specific period is principal when all the payments are of equal value. Given are the interest percentage rate per period (`APR / 12`), the payment period for which the principal portion is desired (`Period`), the total number of payments (`TotPmts`), the present value or principal of the loan (`PVal`), the future value of the loan (`FVal`), and a number that indicates whether the payment is due at the beginning or end of the payment period (`PayType`).

```
Dim NL, TB, Fmt, FVal, PVal, APR, TotPmts, PayType, Payment, Msg, MakeChart, Period, P, I
Const ENDPERIOD = 0, BEGINPERIOD = 1   ' When payments are made.
NL = Chr(13) & Chr(10)   ' Define newline.
TB = Chr(9)   ' Define tab.
Fmt = "###,###,##0.00"   ' Define money format.
FVal = 0   ' Usually 0 for a loan.
PVal = InputBox("How much do you want to borrow?")
APR = InputBox("What is the annual percentage rate of your loan?")
If APR > 1 Then APR = APR / 100   ' Ensure proper form.
TotPmts = InputBox("How many monthly payments do you have to make?")
PayType = MsgBox("Do you make payments at the end of month?", vbYesNo)
If PayType = vbNo Then PayType = BEGINPERIOD Else PayType = ENDPERIOD
Payment = Abs(-Pmt(APR / 12, TotPmts, PVal, FVal, PayType))
Msg = "Your monthly payment is " & Format(Payment, Fmt) & ". "
Msg = Msg & "Would you like a breakdown of your principal and "
Msg = Msg & "interest per period?"
MakeChart = MsgBox(Msg, vbYesNo)   ' See if chart is desired.
If MakeChart <> vbNo Then
   If TotPmts > 12 Then MsgBox "Only first year will be shown."
   Msg = "Month  Payment  Principal  Interest" & NL
   For Period = 1 To TotPmts
      If Period > 12 Then Exit For   ' Show only first 12.
      P = PPmt(APR / 12, Period, TotPmts, -PVal, FVal, PayType)
      P = (Int((P + .005) * 100) / 100)   ' Round principal.
      I = Payment - P
      I = (Int((I + .005) * 100) / 100)   ' Round interest.
      Msg = Msg & Period & TB & Format(Payment, Fmt)
      Msg = Msg & TB & Format(P, Fmt) & TB & Format(I, Fmt) & NL
   Next Period
   MsgBox Msg   ' Display amortization table.
End If
```

Visual Basic for Applications Reference

Visual Studio 6.0

PV Function

See Also Example Specifics

Returns a Double specifying the present value of an annuity based on periodic, fixed payments to be paid in the future and a fixed interest rate.

Syntax

PV(*rate*, *nper*, *pmt*[, *fv*[, *type*]])

The **PV** function has these named arguments:

Part	Description
rate	Required. **Double** specifying interest rate per period. For example, if you get a car loan at an annual percentage rate (APR) of 10 percent and make monthly payments, the rate per period is 0.1/12, or 0.0083.
nper	Required. Integer specifying total number of payment periods in the annuity. For example, if you make monthly payments on a four-year car loan, your loan has a total of 4 * 12 (or 48) payment periods.
pmt	Required. **Double** specifying payment to be made each period. Payments usually contain principal and interest that doesn't change over the life of the annuity.
fv	Optional. Variant specifying future value or cash balance you want after you've made the final payment. For example, the future value of a loan is $0 because that's its value after the final payment. However, if you want to save $50,000 over 18 years for your child's education, then $50,000 is the future value. If omitted, 0 is assumed.
type	Optional. **Variant** specifying when payments are due. Use 0 if payments are due at the end of the payment period, or use 1 if payments are due at the beginning of the period. If omitted, 0 is assumed.

Remarks

An annuity is a series of fixed cash payments made over a period of time. An annuity can be a loan (such as a home mortgage) or an investment (such as a monthly savings plan).

The ***rate*** and ***nper*** arguments must be calculated using payment periods expressed in the same units. For example, if ***rate*** is calculated using months, ***nper*** must also be calculated using months.

For all arguments, cash paid out (such as deposits to savings) is represented by negative numbers; cash received (such as dividend checks) is represented by positive numbers.

Visual Basic for Applications Reference

PV Function Example

In this example, the **PV** function returns the present value of an $1,000,000 annuity that will provide $50,000 a year for the next 20 years. Provided are the expected annual percentage rate (APR), the total number of payments (TotPmts), the amount of each payment (YrIncome), the total future value of the investment (FVal), and a number that indicates whether each payment is made at the beginning or end of the payment period (PayType). Note that YrIncome is a negative number because it represents cash paid out from the annuity each year.

```
Dim Fmt, APR, TotPmts, YrIncome, FVal, PayType, PVal
Const ENDPERIOD = 0, BEGINPERIOD = 1   ' When payments are made.
Fmt = "###,##0.00"   ' Define money format.
APR = .0825   ' Annual percentage rate.
TotPmts = 20   ' Total number of payments.
YrIncome = 50000   ' Yearly income.
FVal = 1000000   ' Future value.
PayType = BEGINPERIOD   ' Payment at beginning of month.
PVal = PV(APR, TotPmts, -YrIncome, FVal, PayType)
MsgBox "The present value is " & Format(PVal, Fmt) & "."
```

Visual Basic for Applications Reference

Visual Studio 6.0

QBColor Function

See Also Example Specifics

Returns a Long representing the RGB color code corresponding to the specified color number.

Syntax

QBColor(*color***)**

The required *color* argument is a whole number in the range 015.

Settings

The *color* argument has these settings:

Number	Color	Number	Color
0	Black	8	Gray
1	Blue	9	Light Blue
2	Green	10	Light Green
3	Cyan	11	Light Cyan
4	Red	12	Light Red
5	Magenta	13	Light Magenta
6	Yellow	14	Light Yellow
7	White	15	Bright White

Remarks

The *color* argument represents color values used by earlier versions of Basic (such as Microsoft Visual Basic for MS-DOS and the Basic Compiler). Starting with the least-significant byte, the returned value specifies the red, green, and blue values used to set the appropriate color in the RGB system used by Visual Basic for Applications.

Visual Basic for Applications Reference

QBColor Function Example

This example uses the **QBColor** function to change the **BackColor** property of the form passed in as `MyForm` to the color indicated by `ColorCode`. **QBColor** accepts integer values between 0 and 15.

```
Sub ChangeBackColor (ColorCode As Integer, MyForm As Form)
   MyForm.BackColor = QBColor(ColorCode)
End Sub
```

Visual Basic for Applications Reference

Visual Studio 6.0

Rate Function

See Also Example Specifics

Returns a Double specifying the interest rate per period for an annuity.

Syntax

Rate(*nper*, *pmt*, *pv*[, *fv*[, *type*[, *guess*]]])

The **Rate** function has these named arguments:

Part	Description
nper	Required. **Double** specifying total number of payment periods in the annuity. For example, if you make monthly payments on a four-year car loan, your loan has a total of 4 * 12 (or 48) payment periods.
pmt	Required. **Double** specifying payment to be made each period. Payments usually contain principal and interest that doesn't change over the life of the annuity.
pv	Required. **Double** specifying present value, or value today, of a series of future payments or receipts. For example, when you borrow money to buy a car, the loan amount is the present value to the lender of the monthly car payments you will make.
fv	Optional. Variant specifying future value or cash balance you want after you make the final payment. For example, the future value of a loan is $0 because that's its value after the final payment. However, if you want to save $50,000 over 18 years for your child's education, then $50,000 is the future value. If omitted, 0 is assumed.
type	Optional. **Variant** specifying a number indicating when payments are due. Use 0 if payments are due at the end of the payment period, or use 1 if payments are due at the beginning of the period. If omitted, 0 is assumed.
guess	Optional. **Variant** specifying value you estimate will be returned by **Rate**. If omitted, ***guess*** is 0.1 (10 percent).

Remarks

An annuity is a series of fixed cash payments made over a period of time. An annuity can be a loan (such as a home mortgage) or an investment (such as a monthly savings plan).

For all arguments, cash paid out (such as deposits to savings) is represented by negative numbers; cash received (such as dividend checks) is represented by positive numbers.

Rate is calculated by iteration. Starting with the value of ***guess***, **Rate** cycles through the calculation until the result is accurate to within 0.00001 percent. If **Rate** can't find a result after 20 tries, it fails. If your guess is 10 percent and **Rate** fails, try a different value for ***guess***.

Visual Basic for Applications Reference

Rate Function Example

This example uses the **Rate** function to calculate the interest rate of a loan given the total number of payments (`TotPmts`), the amount of the loan payment (`Payment`), the present value or principal of the loan (`PVal`), the future value of the loan (`FVal`), a number that indicates whether the payment is due at the beginning or end of the payment period (`PayType`), and an approximation of the expected interest rate (`Guess`).

```
Dim Fmt, FVal, Guess, PVal, Payment, TotPmts, PayType, APR
Const ENDPERIOD = 0, BEGINPERIOD = 1   ' When payments are made.
Fmt = "##0.00"   ' Define percentage format.
FVal = 0   ' Usually 0 for a loan.
Guess = .1   ' Guess of 10 percent.
PVal = InputBox("How much did you borrow?")
Payment = InputBox("What's your monthly payment?")
TotPmts = InputBox("How many monthly payments do you have to make?")
PayType = MsgBox("Do you make payments at the end of the month?", _
vbYesNo)
If PayType = vbNo Then PayType = BEGINPERIOD Else PayType = ENDPERIOD
APR = (Rate(TotPmts, -Payment, PVal, FVal, PayType, Guess) * 12) * 100
MsgBox "Your interest rate is " & Format(CInt(APR), Fmt) & " percent."
```

Visual Basic for Applications Reference

Visual Studio 6.0

Replace Function

See Also Example Specifics

Description

Returns a string in which a specified substring has been replaced with another substring a specified number of times.

Syntax

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

The **Replace** function syntax has these named arguments:

Part	Description
expression	Required. String expression containing substring to replace.
find	Required. Substring being searched for.
replace	Required. Replacement substring.
start	Optional. Position within ***expression*** where substring search is to begin. If omitted, 1 is assumed.
count	Optional. Number of substring substitutions to perform. If omitted, the default value is 1, which means make all possible substitutions.
compare	Optional. Numeric value indicating the kind of comparison to use when evaluating substrings. See Settings section for values.

Settings

The ***compare*** argument can have the following values:

Constant	Value	Description
vbUseCompareOption	1	Performs a comparison using the setting of the **Option Compare** statement.
vbBinaryCompare	0	Performs a binary comparison.
vbTextCompare	1	Performs a textual comparison.

vbDatabaseCompare	2	Microsoft Access only. Performs a comparison based on information in your database.

Return Values

Replace returns the following values:

If	Replace returns
expression is zero-length	Zero-length string ("")
expression is **Null**	An error.
find is zero-length	Copy of ***expression***.
replace is zero-length	Copy of ***expression*** with all occurences of *find* removed.
start > **Len(*expression*)**	Zero-length string.
count is 0	Copy of ***expression***.

Remarks

The return value of the **Replace** function is a string, with substitutions made, that begins at the position specified by ***start*** and and concludes at the end of the ***expression*** string. It is not a copy of the original string from start to finish.

Visual Basic for Applications Reference

Visual Studio 6.0

RGB Function

See Also Example Specifics

Returns a Long whole number representing an RGB color value.

Syntax

RGB(*red*, *green*, *blue*)

The **RGB** function syntax has these named arguments:

Part	Description
red	Required; **Variant** (**Integer**). Number in the range 0255, inclusive, that represents the red component of the color.
green	Required; **Variant** (**Integer**). Number in the range 0255, inclusive, that represents the green component of the color.
blue	Required; **Variant** (**Integer**). Number in the range 0255, inclusive, that represents the blue component of the color.

Remarks

Application methods and properties that accept a color specification expect that specification to be a number representing an RGB color value. An RGB color value specifies the relative intensity of red, green, and blue to cause a specific color to be displayed.

The value for any argument to **RGB** that exceeds 255 is assumed to be 255.

The following table lists some standard colors and the red, green, and blue values they include:

Color	Red Value	Green Value	Blue Value
Black	0	0	0
Blue	0	0	255
Green	0	255	0
Cyan	0	255	255

Red	255	0	0
Magenta	255	0	255
Yellow	255	255	0
White	255	255	255

RGB Function Example

This content is no longer actively maintained. It is provided as is, for anyone who may still be using these technologies, with no warranties or claims of accuracy with regard to the most recent product version or service release.

This example sets the gridline color in the active window in Book1.xls to red.

```
Workbooks("BOOK1.XLS").Worksheets("Sheet1").Activate
ActiveWindow.GridlineColor = RGB(255,0,0)
```

This example sets the color of the tick labels on the value axis in Chart1 to green.

```
Charts("Chart1").Axes(xlValue).TickLabels _
    .Font.Color = RGB(0, 255, 0)
```

This example sets the marker background and foreground colors for the second point in series one in Chart1 to green and red, respectively.

```
With Charts("Chart1").SeriesCollection(1).Points(2)
    .MarkerBackgroundColor = RGB(0,255,0)    ' green
    .MarkerForegroundColor = RGB(255,0,0)    ' red
End With
```

This example sets the interior pattern color for rectangle one on Sheet1 to red.

```
With Worksheets("Sheet1").Rectangles(1).Interior
    .Pattern = xlGrid
    .PatternColor = RGB(255,0,0)
End With
```

Visual Basic for Applications Reference

Visual Studio 6.0

Right Function

See Also Example Specifics

Returns a **Variant** (**String**) containing a specified number of characters from the right side of a string.

Syntax

Right(***string***, ***length***)

The **Right** function syntax has these named arguments:

Part	Description
string	Required. String expression from which the rightmost characters are returned. If ***string*** contains Null, **Null** is returned.
length	Required; **Variant** (**Long**). Numeric expression indicating how many characters to return. If 0, a zero-length string ("") is returned. If greater than or equal to the number of characters in ***string***, the entire string is returned.

Remarks

To determine the number of characters in ***string***, use the **Len** function.

Note Use the **RightB** function with byte data contained in a string. Instead of specifying the number of characters to return, ***length*** specifies the number of bytes.

Visual Basic for Applications Reference

Right Function Example

This example uses the **Right** function to return a specified number of characters from the right side of a string.

```
Dim AnyString, MyStr
AnyString = "Hello World"   ' Define string.
MyStr = Right(AnyString, 1)   ' Returns "d".
MyStr = Right(AnyString, 6)   ' Returns " World".
MyStr = Right(AnyString, 20)   ' Returns "Hello World".
```

Visual Basic for Applications Reference

Visual Studio 6.0

Rnd Function

See Also Example Specifics

Returns a **Single** containing a random number.

Syntax

Rnd[(*number*)]

The optional *number* argument is a Single or any valid numeric expression.

Return Values

If *number* is	**Rnd generates**
Less than zero	The same number every time, using *number* as the seed.
Greater than zero	The next random number in the sequence.
Equal to zero	The most recently generated number.
Not supplied	The next random number in the sequence.

Remarks

The **Rnd** function returns a value less than 1 but greater than or equal to zero.

The value of *number* determines how **Rnd** generates a random number:

For any given initial seed, the same number sequence is generated because each successive call to the **Rnd** function uses the previous number as a seed for the next number in the sequence.

Before calling **Rnd,** use the **Randomize** statement without an argument to initialize the random-number generator with a seed based on the system timer.

To produce random integers in a given range, use this formula:

```
Int((upperbound - lowerbound + 1) * Rnd + lowerbound)
```

Here, *upperbound* is the highest number in the range, and *lowerbound* is the lowest number in the range.

Note To repeat sequences of random numbers, call **Rnd** with a negative argument immediately before using **Randomize** with a numeric argument. Using **Randomize** with the same value for *number* does not repeat the previous sequence.

Security Note Because the **Random** statement and the **Rnd** function start with a seed value and generate numbers that fall within a finite range, the results may be predictable by someone who knows the algorithm used to generate them. Consequently, the **Random** statement and the **Rnd** function should not be used to generate random numbers for use in cryptography.

Visual Basic for Applications Reference

Rnd Function Example

This example uses the **Rnd** function to generate a random integer value from 1 to 6.

```
Dim MyValue
MyValue = Int((6 * Rnd) + 1)   ' Generate random value between 1 and 6.
```

Visual Basic for Applications Reference

Visual Studio 6.0

Round Function

See Also Example Specifics

Description

Returns a number rounded to a specified number of decimal places.

Syntax

Round(*expression* [**,***numdecimalplaces*]**)**

The **Round** function syntax has these parts:

Part	Description
expression	Required. Numeric expression being rounded.
numdecimalplaces	Optional. Number indicating how many places to the right of the decimal are included in the rounding. If omitted, integers are returned by the **Round** function.