1. Data Type Summary (Visual Basic)

Visual Basic type	Common language runtime type structure	Nominal storage allocation	Value range
Boolean	Boolean	Depends on implementing platform	True or False
Byte	Byte	1 byte	0 through 255 (unsigned)
Char (single character)	Char	2 bytes	0 through 65535 (unsigned)
Date	DateTime	8 bytes	0:00:00 (midnight) on January 1, 0001 through 11:59:59 PM on December 31, 9999
Decimal	Decimal	16 bytes	0 through +/- 79,228,162,514,264,337,593,543,950,335 (+/- 7.9...E+28) † with no decimal point; 0 through +/-7.9228162514264337593543950335 with 28 places to the right of the decimal; smallest nonzero number is +/- 0.0000000000000000000000000001 (+/-1E-28) †
Double (double-precision floating-point)	Double	8 bytes	-1.79769313486231570E+308 through -4.94065645841246544E-324 † for negative values; 4.94065645841246544E-324 through 1.79769313486231570E+308 † for positive values
Integer	Int32	4 bytes	-2,147,483,648 through 2,147,483,647 (signed)
Long (long integer)	Int64	8 bytes	-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807 (9.2...E+18 †) (signed)
Object	Object (class)	4 bytes on 32-bit platform 8 bytes on 64-bit platform	Any type can be stored in a variable of type Object
SByte	SByte	1 byte	-128 through 127 (signed)
Short (short integer)	Int16	2 bytes	-32,768 through 32,767 (signed)
Single (single-precision floating-	Single	4 bytes	-3.4028235E+38 through -1.401298E-45 † for negative values; 1.401298E-45 through 3.4028235E+38 † for positive values

point)			
String (variable-length)	String (class)	Depends on implementing platform	0 to approximately 2 billion Unicode characters
UInteger	UInt32	4 bytes	0 through 4,294,967,295 (unsigned)
ULong	UInt64	8 bytes	0 through 18,446,744,073,709,551,615 (1.8...E+19 †) (unsigned)
User-Defined (structure)	(inherits from ValueType)	Depends on implementing platform	Each member of the structure has a range determined by its data type and independent of the ranges of the other members
UShort	UInt16	2 bytes	0 through 65,535 (unsigned)

2. Conversion Functions (Visual Basic)

2.1. Conversion Functions (Visual Basic)

Asc
AscW
CBool Function
CByte Function
CChar Function
CDate Function
CDbl Function
CDec Function
Chr
ChrW
CInt Function
CLng Function
CObj Function
CSByte Function
CShort Function
CSng Function
CStr Function
CType Function
CUInt Function
CULng Function
CUShort Function
Format
Hex
Oct
Str
Val

2.2. Math

.NET Framework method	Description
Abs	Returns the absolute value of a number.
Acos	Returns the angle whose cosine is the specified number.
Asin	Returns the angle whose sine is the specified number.
Atan	Returns the angle whose tangent is the specified number.
Atan2	Returns the angle whose tangent is the quotient of two specified numbers.
BigMul	Returns the full product of two 32-bit numbers.
Ceiling	Returns the smallest integral value that's greater than or equal to the specified Decimal or Double.
Cos	Returns the cosine of the specified angle.
Cosh	Returns the hyperbolic cosine of the specified angle.
DivRem	Returns the quotient of two 32-bit or 64-bit signed integers, and also returns the remainder in an output parameter.
Exp	Returns e (the base of natural logarithms) raised to the specified power.
Floor	Returns the largest integer that's less than or equal to the specified Decimal or Double number.
IEEERemainder	Returns the remainder that results from the division of a specified number by another specified number.
Log	Returns the natural (base e) logarithm of a specified number or the logarithm of a specified number in a specified base.
Log10	Returns the base 10 logarithm of a specified number.
Max	Returns the larger of two numbers.
Min	Returns the smaller of two numbers.
Pow	Returns a specified number raised to the specified power.
Round	Returns a Decimal or Double value rounded to the nearest integral value or to a specified number of fractional digits.
Sign	Returns an Integer value indicating the sign of a number.
Sin	Returns the sine of the specified angle.
Sinh	Returns the hyperbolic sine of the specified angle.
Sqrt	Returns the square root of a specified number.
Tan	Returns the tangent of the specified angle.
Tanh	Returns the hyperbolic tangent of the specified angle.
Truncate	Calculates the integral part of a specified Decimal or Double number.

3. String Functions (Visual Basic)

.NET Framework method	Description
Asc, AscW	Returns an Integer value representing the character code corresponding to a character.
Chr, ChrW	Returns the character associated with the specified character code.
Filter	Returns a zero-based array containing a subset of a String array based on specified filter criteria.
Format	Returns a string formatted according to instructions contained in a format String expression.
FormatCurrency	Returns an expression formatted as a currency value using the currency symbol defined in the system control panel.
FormatDateTime	Returns a string expression representing a date/time value.
FormatNumber	Returns an expression formatted as a number.
FormatPercent	Returns an expression formatted as a percentage (that is, multiplied by 100) with a trailing % character.
InStr	Returns an integer specifying the start position of the first occurrence of one string within another.
InStrRev	Returns the position of the first occurrence of one string within another, starting from the right side of the string.
Join	Returns a string created by joining a number of substrings contained in an array.
LCase	Returns a string or character converted to lowercase.
Left	Returns a string containing a specified number of characters from the left side of a string.
Len	Returns an integer that contains the number of characters in a string.
LSet	Returns a left-aligned string containing the specified string adjusted to the specified length.
LTrim	Returns a string containing a copy of a specified string with no leading spaces.
Mid	Returns a string containing a specified number of characters from a string.
Replace	Returns a string in which a specified substring has been replaced with another substring a specified number of times.
Right	Returns a string containing a specified number of characters from the right side of a string.
RSet	Returns a right-aligned string containing the specified string adjusted to the specified length.
RTrim	Returns a string containing a copy of a specified string with no trailing spaces.
Space	Returns a string consisting of the specified number of spaces.
Split	Returns a zero-based, one-dimensional array containing a specified number of substrings.
StrComp	Returns -1, 0, or 1, based on the result of a string comparison.
StrConv	Returns a string converted as specified.
StrDup	Returns a string or object consisting of the specified character

	repeated the specified number of times.
StrReverse	Returns a string in which the character order of a specified string is reversed.
Trim	Returns a string containing a copy of a specified string with no leading or trailing spaces.
UCase	Returns a string or character containing the specified string converted to uppercase.

4. Type Conversion Functions (Visual Basic)

```
CBool(expression)
CByte(expression)
CChar(expression)
CDate(expression)
CDbl(expression)
CDec(expression)
CInt(expression)
CLng(expression)
CObj(expression)
CSByte(expression)
CShort(expression)
CSng(expression)
CStr(expression)
CUInt(expression)
CULng(expression)
CUShort(expression)
```

5. Arithmetic Operators (Visual Basic)

^ Operator

* Operator

/ Operator

\ Operator

Mod Operator

+ Operator (unary and binary)

- Operator (unary and binary)

6. Assignment Operators (Visual Basic)

The following are the assignment operators defined in Visual Basic.

= Operator

^= Operator

*= Operator

/= Operator

\= Operator

+= Operator

-= Operator

<<= Operator

>>= Operator

&= Operator

6.1. Bit Shift Operators (Visual Basic)

The following are the bit shift operators defined in Visual Basic.
<< Operator
>> Operator

6.2. Comparison Operators

< operator
<= operator
> operator
>= operator
= operator
<> operator
Is Operator (Visual Basic)
IsNot Operator (Visual Basic)
Like Operator (Visual Basic)

6.3. Concatenation Operators

& Operator
+ Operator

6.4. Logical/Bitwise Operators

And Operator (Visual Basic)
Not Operator (Visual Basic)
Or Operator (Visual Basic)
Xor Operator (Visual Basic)
AndAlso Operator (Visual Basic)
OrElse Operator (Visual Basic)
IsFalse Operator (Visual Basic)
IsTrue Operator (Visual Basic)

7. Miscellaneous Operators

AddressOf Operator (Visual Basic)
Await Operator (Visual Basic)
GetType Operator (Visual Basic)
Function Expression (Visual Basic)
If Operator (Visual Basic)
TypeOf Operator (Visual Basic)

8. Statement

9. For...Next Statement

Repeats a group of statements a specified number of times.

```
For counter [ As datatype ] = start To end [ Step step ]
    [ statements ]
    [ Continue For ]
    [ statements ]
    [ Exit For ]
    [ statements ]
Next [ counter ]
```

Part	Description
counter	Required in the For statement. Numeric variable. The control variable for the loop. For more information, see Counter Argument later in this topic.
datatype	Optional. Data type of counter. For more information, see Counter Argument later in this topic.
start	Required. Numeric expression. The initial value of counter.
end	Required. Numeric expression. The final value of counter.
step	Optional. Numeric expression. The amount by which counter is incremented each time through the loop.
statements	Optional. One or more statements between For and Next that run the specified number of times.
Continue For	Optional. Transfers control to the next loop iteration.
Exit For	Optional. Transfers control out of the For loop.
Next	Required. Terminates the definition of the For loop.

9.1. Do…Loop Statement

Repeats a block of statements while a Boolean condition is True or until the condition becomes True.

```
Do { While | Until } condition
    [ statements ]
    [ Continue Do ]
    [ statements ]
    [ Exit Do ]
    [ statements ]
Loop
-or-
Do
    [ statements ]
    [ Continue Do ]
    [ statements ]
    [ Exit Do ]
    [ statements ]
Loop { While | Until } condition
```

Term	Definition
Do	Required. Starts the definition of the Do loop.
While	Required unless Until is used. Repeat the loop until condition is False.
Until	Required unless While is used. Repeat the loop until condition is True.
condition	Optional. Boolean expression. If condition is Nothing, Visual Basic treats it as False.
statements	Optional. One or more statements that are repeated while, or until, condition is True.
Continue Do	Optional. Transfers control to the next iteration of the Do loop.
Exit Do	Optional. Transfers control out of the Do loop.
Loop	Required. Terminates the definition of the Do loop.

Use a Do...Loop structure when you want to repeat a set of statements an indefinite number of times, until a condition is satisfied. If you want to repeat the statements a set number of times, the For...Next Statement is usually a better choice.

9.2. While…End While Statement

Runs a series of statements as long as a given condition is True.

```
While condition
    [ statements ]
    [ Continue While ]
    [ statements ]
    [ Exit While ]
    [ statements ]
End While
```

Term	Definition
condition	Required. Boolean expression. If condition is Nothing, Visual Basic treats it as False.
statements	Optional. One or more statements following While, which run every time condition is True.
Continue While	Optional. Transfers control to the next iteration of the While block.
Exit While	Optional. Transfers control out of the While block.
End While	Required. Terminates the definition of the While block.

Use a While...End While structure when you want to repeat a set of statements an indefinite number of times, as long as a condition remains

10. Try…Catch

Provides a way to handle some or all possible errors that may occur in a given block of code, while still running code.

```
Try
    [ tryStatements ]
    [ Exit Try ]
[ Catch [ exception [ As type ] ] [ When expression ]
    [ catchStatements ]
    [ Exit Try ] ]
[ Catch ... ]
[ Finally
    [ finallyStatements ] ]
End Try
```

Term	Definition
tryStatements	Optional. Statement(s) where an error can occur. Can be a compound statement.
Catch	Optional. Multiple Catch blocks permitted. If an exception occurs when processing the Try block, each Catch statement is examined in textual order to determine whether it handles the exception, with exception representing the exception that has been thrown.
exception	Optional. Any variable name. The initial value of exception is the value of the thrown error. Used with Catch to specify the error caught. If omitted,

	the Catch statement catches any exception.
type	Optional. Specifies the type of class filter. If the value of exception is of the type specified by type or of a derived type, the identifier becomes bound to the exception object.
When	Optional. A Catch statement with a When clause catches exceptions only when expression evaluates to True. A When clause is applied only after checking the type of the exception, and expression may refer to the identifier representing the exception.
expression	Optional. Must be implicitly convertible to Boolean. Any expression that describes a generic filter. Typically used to filter by error number. Used with When keyword to specify circumstances under which the error is caught.
catchStatements	Optional. Statement(s) to handle errors that occur in the associated Try block. Can be a compound statement.
Exit Try	Optional. Keyword that breaks out of the Try...Catch...Finally structure. Execution resumes with the code immediately following the End Try statement. The Finally statement will still be executed. Not allowed in Finally blocks.
Finally	Optional. A Finally block is always executed when execution leaves any part of the Try...Catch statement.
finallyStatements	Optional. Statement(s) that are executed after all other error processing has occurred.
End Try	Terminates the Try...Catch...Finally structure.

If you expect that a particular exception might occur during a particular section of code, put the code in a Try block and use a Catch block to retain control and handle the exception if it occurs.