Appendix C: dBASE Expressions

In CodeBase, a dBASE expression is represented as a character string and is evaluated using the expression evaluation functions. dBASE expressions are used to define the keys and filters of an index file. They can be useful for other purposes such as interactive queries.

The dBASE functions listed in this appendix are not C functions to be called directly from C programs. They are dBASE functions that are recognized by the CodeBase expression evaluation functions. In the same manner, C functions and variables cannot appear in an dBASE expression.

General dBASE Expression Information

All dBASE expressions return a value of a specific type. This type can be Numeric, Character, Date or Logical. A common form of a dBASE expression is the name of a field. In this case, the type of the dBASE expression is the type of the field. Field names, constants, and functions can all be used as parts of a dBASE expression. These parts can be combined with other functions or with operators.

Example dBASE Expression: "FIELD_NAME"

Memo fields evaluate to a maximum length as determined by the setting of Code4::memSizeMemoExpr.

Note In this manual all dBASE expressions are contained in double quotes (" "). The quotes are not considered part of the dBASE expression. Any double quotes that are contained within the dBASE expression will be denoted as ' \" '. This method is used to remain consistent with the format of C++ string constants.

Field Name Qualifier

It is possible to qualify a field name in a dBASE expression by specifying the data file.

Example dBASE Expression: "DBALIAS->FLD_NAME"

Observe that the first part the qualifier specifies a data file alias (see Data4::.alias). This is usually just the name of the data file. Then there is the "->" followed by the field name.

dBASE Expression Constants

dBASE Expressions can consist of a Numeric, Character or Logical constant. However, dBASE expressions that are constants are usually not very useful. Constants are usually used within a more complicated dBASE expression.

A Numeric constant is a number. For example, "5", "7.3", and "18" are all dBASE expressions containing Numeric constants.

Character constants are letters with quote marks around them. " 'This is data' ", " 'John Smith' ", and " \"John Smith\" " are all examples of dBASE expressions containing Character constants. If you wish to specify a character constant with a single quote or a double quote contained inside it, use the other type of quote to mark the Character constant. For example," \"Man's\" " and " ' \"Ok\" ' " are both legitimate Character constants. Unless otherwise

specified, all dBASE Character constants in this manual are denoted by single quote characters.

Constants .TRUE. and .FALSE. are the only legitimate Logical constants. Constants .T. and .F. are legitimate abbreviations.

dBASE Expression Operators

dBASE Operators like '+' , ' * ', or '<' are used to manipulate constants and fields. For example, "3+8" is an example of a dBASE expression in which the Add operator acts on two Numeric constants to return the Numeric value "11". The values an operator acts on must have a type appropriate for the operator. For example, the divide '/' operator acts on two Numeric values.

Precedence Operators have a precedence that specifies operator evaluation order. The precedence of each operator is specified in the following tables that describe the various operators. The higher the precedence, the earlier the operation will be performed. For example, 'divide' has a precedence of 6 and 'plus' has a precedence of 5 which means 'divide' is evaluated before 'plus'. Consequently, "1+4/2" is "3". Evaluation order can be made explicit by using brackets. For example, "1+2 * 3" returns "7" and "(1+2) * 3" returns "9".

Numeric Operators The Numeric operators all operate on Numeric values.

Operator Name	Symbol	Precedence
Add	+	5
Subtract	-	5
Multiply	*	6
Divide	/	6
Exponent	** or ^	7

Character Operators There are two Character operators, named "Concatenate I" and "Concatenate II", which combine two Character values into one. They are distinguished from the Add and Subtract operators by the types of the values they operate on.

Operator Name	Symbol	Precedence
Concatenate I	+	5
Concatenate II	-	5

Examples: " 'John ' + 'Smith' " becomes " 'John Smith' "

" 'ABC' + 'DEF' " becomes " 'ABCDEF' "

Concatenate II is slightly different in that any spaces at the end of the first Character value are moved to the end of the result.

" 'John'-'Smith ' " becomes " 'JohnSmith ' "

" 'ABC' - 'DEF' " becomes " 'ABCDEF' "

" 'A ' - 'D ' " becomes " 'AD ' "

Relational Operators Relational Operators are operators that return a Logical result (which is either true or false). All operators, except Contain, operate on Numeric, Character or Date values. Contain operates on two character values and returns true if the first is contained in the second.

Operator Name	Symbol	Precedence
Equal To	=	4
Not Equal To	<> or #	4

Less Than	<	4
Greater Than	>	4
Less Than or Equal To	< =	4
Greater Than or Equal To	> =	4
Contain	$	4

Examples: " 'CD' $ 'ABCD' " returns ".T."
" 8<7 " returns ".F."

Logical Operators Logical Operators return a Logical Result and operate on two Logical values.

Operator Name	Symbol	Precedence
Not	.NOT.	3
And	.AND.	2
Or	.OR.	1

Examples " .NOT. .T. " returns ".F."
" .T. .AND. .F." returns ".F."

dBASE Expression Functions

A function can be used as a dBASE expression or as part of an dBASE expression. Like operators, constants, and fields, dBase functions return a value. Functions always have a function name and are followed by a left and right parentheses. Some functions take parameters within the parentheses.

ALLTRIM(CHAR_VALUE)

This function trims all of the blanks from both the beginning and the end of the expression.

ASCEND(VALUE)

This function is not supported by dBASE, FoxPro or Clipper.

ASCEND() accepts all types of parameters, except complex Numeric expressions. ASCEND() converts all types into a Character type in ascending order. In the case of Numeric types, the conversion is done so that the sorting will work correctly even if negative values are present.

CHR(INTEGER_VALUE)

This function returns the character whose numeric ASCII code is identical to the given integer. The integer must be between 0 and 255.

Example: CHR(65) returns A.

CTOD(CHAR_VALUE)

The function converts a Character value into a Date value.

e.g. " CTOD("11/30/88") "

The character representation is always in the format specified by the Code4::dateFormat member variable which is by default "MM/DD/YY".

DATE()

The system date is returned.

DATETIME(YEAR, MONTH, DAY [, HOUR [, MINUTE [, SECOND]]])

The given date and time is returned as a DateTime.

e.g. "STOPTIME = DATETIME(2003,03,21,13,30,00)"

This expression would evaluate to .TRUE. if the DateTime field STOPTIME is set to March 21, 2003 at 1:30 pm.

DAY(DATE_VALUE)

Returns the day of the Date parameter as a Numeric value from "1" to

"31".
e.g. "DAY(DATE())"
Returns "30" if it is the thirtieth of the month.

DESCEND(VALUE)

This function is not supported by dBASE or FoxPro. DESCEND() is compatible with Clipper, only if the parameter is a Character type.

DESCEND() accepts any type of parameter, except complex numeric expressions. DESCEND() converts all types into a character type in descending order.
For example, the following expression would produce a reverse order
sort on the field ORD_DATE followed by normal sub-sort on COMPANY.
e.g. DESCEND(ORD_DATE) + COMPANY
See also ASCEND().

DELETED()

Returns .TRUE. if the current record is marked for deletion.

DTOC(DATE_VALUE) DTOC(DATE_VALUE, 1)

The function converts a Date value into a Character value. The format
of the resulting Character value is specified by the Code4::dateFormat member variable which is by default "MM/DD/YY".
e.g. " DTOC(DATE()) "
Returns the Character value "05/30/87" if the date is May 30, 1987.

If the optional second argument is used, the result will be identical to the dBASE expression function DTOS.
For example, DTOC(DATE(), 1) will return "19940731" if the date is July 31, 1994.

DTOS(DATE_VALUE)

The function converts a Date value into a Character value. The format of the resulting Character value is "CCYYMMDD". e.g. " DTOS(DATE()) "
Returns the Character value "19870530" if the date is May 30, 1987.

IIF(LOG_VALUE, TRUE_RESULT, FALSE_RESULT)

If Log_Value is .TRUE. then IIF returns the True_Result value. Otherwise, IIF returns the False_Result value. Both True_Result and False_Result must be the same length and type. Otherwise, an error results.
e.g. "IIF(VALUE < 0, "Less than zero ", "Greater than zero")"

e.g. "IIF(NAME = "John", "The name is John", "Not John ")"

LEFT(CHAR_VALUE, NUM_CHARS)

This function returns a specified number of characters from a Character expression, beginning at the first character on the left. The parameter NUM_CHARS must be constant.
e.g. "LEFT('SEQUITER', 3)" returns "SEQ".
The same result could be achieved with "SUBSTR ('SEQUITER', 1, 3)".

LTRIM(CHAR_VALUE)

This function trims any blanks from the beginning of the expression.

MONTH(DATE_VALUE)

Returns the month of the Date parameter as a Numeric.
e.g. " MONTH(DT_FIELD) "
Returns 12 if the Date field's month is December.

PAGENO()

When using the report module or CodeReporter, this function returns the current report page number.

RECCOUNT()

The record count function returns the total number of records in the database:
e.g. " RECCOUNT() "
Returns 10 if there are ten records in the database.

RECNO()

The record number function returns the record number of the current record.

RIGHT(CHAR_VALUE, NUM_CHARS)

This function returns a specified number of characters from the end of a character expression. The parameter 'NUM_CHARS' must be constant.
e.g. "RIGHT('SEQUITER', 3)" returns "TER".

STOD(CHAR_VALUE)

The function converts a Character value into a Date value: e.g. " STOD('19881130') "
The character representation is in the format "CCYYMMDD".

STR(NUMBER, LENGTH, DECIMALS)

The function converts a Numeric value into a Character value. Length is the number of characters in the new string, including the decimal point. Decimals is the number of decimal places desired. The parameters LENGTH and DECIMALS must be constant. If the

number is too big for the allotted space, *'s will be returned.
e.g. " STR(5.7, 4, 2) " returns " '5.70' "

The number 5.7 is converted to a string of length 4. In addition, there will be 2 decimal places.

e.g. " STR(5.7, 3, 2) " returns " '***' "
The number 5.7 cannot fit into a string of length 3 if it is to have 2 decimal places. Consequently, *'s are filled in.

SUBSTR(CHAR_VALUE, START_POSITION, NUM_CHARS)

A substring of the Character value is returned. The substring will be NUM_CHARS long, and will start at the START_POSITION character
of CHAR_VALUE. The parameters START_POSITION and NUM_CHARS must be constant.

e.g. " SUBSTR("ABCDE", 2, 3)" returns " 'BCD' "
e.g. "SUBSTR("Mr. Smith", 5, 1)" returns " 'S' "

TIME()

The function returns the system time as a character representation. It uses the following format: HH:MM:SS.

e.g. " TIME() " returns " 12:00:00 " if it is noon.
e.g. " TIME() " returns " 13:30:00 " if it is one thirty PM.

TRIM(CHAR_VALUE)

This function trims any blanks off the end of the expression.

UPPER(CHAR_VALUE)

A Character string is converted to uppercase and the result is returned.

VAL(CHAR_VALUE)

The function converts a Character value to a Numeric value.

e.g. VAL('10') returns "10".
e.g. VAL('-8.7') returns "-8.7".

YEAR(DATE_VALUE)

Returns the year of the date parameter as a Numeric:
e.g. "YEAR(STOD('19920830')) " returns " 1992"

Additional Expressions

More convenient functions have been created for various use:

EMPTY(field or value)
Returns TRUE if the field or value is empty. Returns false if it contains a value.

- characters - returns true if the result contains only blanks
- numbers - returns true if the value is 0
- logicals - returns true if the value is false
- dates/datetimes - returns true if the value resolves to zero (field left blank)

SPACE(numSpaces)
Creates a string containing the input numerical number of blanks. For example SPACE(3) creates a string containing 3 blank spaces (" ").

STRZERO(value, length, decimals)
Identical to STR() except any blanks to the left of the number are replaced by zeros.

Creates a string containing the input numerical value formatted to be of the input numerical length and the optional input numerical decimals. If the optional decimals parameter is left out, the default of no decimals is used.

Examples:

- STRZERO(100.03, 10) creates string "0000000100"
- STRZERO(100, 2) causes an overflow and creates string "**"
- STRZERO(.03, 4) creates the string "0000" since no decimals are specified
- STRZERO(.0004, 3) creates the string “000” since no decimals are specified
- STRZERO(1.44, 2) truncates to "01" because no decimals are specified
- STRZERO(100.03, 10, 4) creates string "00100.0300"
- STRZERO(.0004, 3, 2) truncates to the string ".00"
- STRZERO(0, 6, 3) creates the string "00.000"

DATETIME(year, month, day, hours, minutes, seconds, milliseconds)
Creates a constant datetime output suitable for use with the datetime and datetimeMilli field types. The choice of parameters determines whether the return type is of type datetime or type datetimeMilli. It is important that you create an expression of the same type as the field type to avoid type incompatibility errors.

For the datetime type, use the following numerical parameters:

- year - required
- month - required
- day - required
- hours - optional
- minutes - optional
- seconds - optional

For the datetimeMilli type, use the following numerical parameters:

- year - required
- month - required
- day - required
- hours - required, but can be 0
- minutes - required, but can be 0
- seconds - required, but can be 0
- milliseconds - required, but can be 0

For comparison purposes, you can compare 2 identical types (datetime to datetime), but cannot compare 2 differing types (datetime to datetimeMilli).

DEL()
Creates a 1 character string containing the contents of the deleted flag for the current record. Note that this differs from DELETED() which returns a true/false flag whether or not the record is deleted. An asterisk character '*' is used in the datafile to indicate a record is deleted.

PADL(string, length)
Creates a new string using the input string and the constant numerical length to pad out the string on the left. The final length of the string is the input constant numerical length.

Examples

- PADL("ABC", 4) creates the string " ABC" - 1 blank padded on the left
- PADL("ABCD", 3) creates the string "ABC" - trims due to input length

PADL(TRIM("ABC "), 5) where the trimmed string contains 3 blank trailing characters creates the string " ABC" - 2 blanks padded on the left

PADR(string, length)
Creates a new string using the input string and the constant numerical length to pad out the string on the right. The final length of the string is the input constant numerical length.

Examples

· PADR("ABC", 4) creates the string "ABC " - 1 blank padded on the right

· PADR("ABCD", 3) creates the string "ABC" - trims due to input length

· PADR(LTRIM(" ABC"), 5) where the trimmed string contains 3 blank preceding characters creates the string "ABC " - 2 blanks padded on the right