bcParser

bcParser

Math Parser for Delphi and C++ Builder

by Bestcode.com

bcParser Math Parser for Delphi and C++ Builder is a VCL component to parse and evaluate mathematical expressions given as strings at runtime.

bcParser allows the user define their own functions and variables.

Typical users of bcParser are financial, engineering, scientific applications.

bcParser

Publisher

Bestcode.com

Special thanks to:

My wife who puts up with me as I spend endless hours on my computer.

Table of Contents

Foreword

My cousin asked me for a function that would evaluate a mathematical expression given as a string. I told him that was easy and I set out to write it. Pretty soon I realized that was not as easy as I hoped. He solved his problem some other way as I could not write it that quick.

That was late 1990s. Then as I ran into more and more situations where I needed an expression parser, I wrote one for myself in Delphi and eventually ported it into multiple languages as I worked in other projects. This manual is for the Delphi version. However, Bestcode.com has parsers for C++, COM, Java, and .NET.

I hope bcParser will be useful to you. Please report the bugs and your wishes to Bestcode.com

Thank you.

Part I

1 bcParser

TbcParser
Mathematical Parser - VCL Component for Delphi and C++ Builder

This help file contains information on TbcParser VCL component.

TbcParser implements parse once-evaluate many times type of parsing for mathematical expressions given as strings.

TbcParser is especially useful in scientific, engineering programs.

It allows creation of custom variables and functions to be used in the expressions.
It's internal arithmetic uses 'Extended' storage for floating point numbers.

For purchasing and support please visit http://www.bestcode.com

Please send your suggestions, comments or bug reports to support@bestcode.com

Units
bcParser

1.1 bcParser unit

TbcParser
Mathematical Parser - VCL Component for Delphi and C++ Builder

This help file contains information on TbcParser VCL component.

TbcParser implements parse once-evaluate many times type of parsing for mathematical expressions given as strings.

TbcParser is especially useful in scientific, engineering programs.

It allows creation of custom variables and functions to be used in the expressions.
It's internal arithmetic uses 'Extended' storage for floating point numbers.

For purchasing and support please visit http://www.bestcode.com

Please send your suggestions, comments or bug reports to support@bestcode.com

USES

SysUtils, Classes

TYPES

NUMBER

EbcParserError
TTwoParamFunc
TOneParamFunc
TNParamFunc
TbcParser

1.2 TYPES

Unit
bcParser

Declaration
TbcParser = class (TComponent)

Description

This mathematical expression parser component parses and evaluates a mathematical expression that may contain variables and functions.

Examples to typical expressions are:

'SIN(3.14)+5^2+POW(2,7)-MAX(10,20)'

'([SIN(X)^2] + COS(1)) * PI/2'

'(COSH(2.71)+ SINH(COS(0.5) * POW(0.2345, 0.67))) - LOG(1-X^2+X^5)'

'(90+292*POW(X,0.31))*(1+0.025*LOG(Y))*(1-1*POW(Y+X,1.09))'

'25993.894*EXP(-1.53389*(X-32))*POW(X, 0.13547)*POW(Y, 0.38603)*EXP(-0.36222*Y)'

The user can add/remove his/her own custom variables and functions to be used in the expression. To be efficient in repeated calculations, parser creates a parse tree at first and reuses this parse tree for each evaluation without the need to reparse.

If Optimization is on, the parse tree will be optimized by calculating constant expression sections at once so that further evaluation requests will be quicker.

1.2.1 NUMBER

Unit
bcParser

Declaration

NUMBER = extended ;

1.2.2 EbcParserError

Unit
bcParser

Declaration
EbcParserError = class (Exception)

1.2.3 TTwoParamFunc

Unit
bcParser

Declaration
TTwoParamFunc = function (const x : NUMBER ;const y : NUMBER) : NUMBER ;

Description
TTwoParamFunc type specifies the prototype of the functions that users can add to the list of available functions with two parameters to be used in an expression.

1.2.4 TOneParamFunc

Unit
bcParser

Declaration
TOneParamFunc = function (const x : NUMBER) : NUMBER ;

Description
TOneParamFunc type specifies the prototype of the functions that users can add to the list of available functions with one parameter to be used in an expression.

1.2.5 TNParamFunc

Unit
bcParser

Declaration
TNParamFunc = function (const x : array of NUMBER) : NUMBER ;

Description
TNParamFunc type specifies the prototype of the functions that users can add to the list of available functions with n number of parameters to be used in an expression.

1.2.6 TbcParser

Unit
bcParser

Declaration
TbcParser = class (TComponent)

Description

This mathematical expression parser component parses and evaluates a mathematical expression that may contain variables and functions.

Examples to typical expressions are:

'SIN(3.14)+5^2+POW(2,7)-MAX(10,20)'

'([SIN(X)^2] + COS(1)) * PI/2'

'(COSH(2.71)+ SINH(COS(0.5) * POW(0.2345, 0.67))) - LOG(1-X^2+X^5)'

'(90+292*POW(X,0.31))*(1+0.025*LOG(Y))*(1-1*POW(Y+X,1.09))'

'25993.894*EXP(-1.53389*(X-32))*POW(X, 0.13547)*POW(Y, 0.38603)*EXP(-0.36222*Y)'

The user can add/remove his/her own custom variables and functions to be used in the expression. To be efficient in repeated calculations, parser creates a parse tree at first and reuses this parse tree for each evaluation without the need to reparse.

If Optimization is on, the parse tree will be optimized by calculating constant expression sections at once so that further evaluation requests will be quicker.

1.2.6.2 Create

Unit
bcParser

Applies to
TbcParser

Declaration
constructor Create (AOwner : TComponent) ;override;

1.2.6.3 Destroy

Unit
bcParser

Applies to
TbcParser

Declaration
destructor Destroy ;override;

1.2.6.4 Parse

Unit
bcParser

Applies to
TbcParser

Declaration
procedure Parse ;virtual;

Description
Parses the expression and forms a parse tree. Throws EbcParserError exception if it cannot parse. Upon successful completion of parsing, it will set the Dirty flag to false, so that unless the expression is changed it does not need to re-parsed. Users may want to call the parse method directly to check the validity of an input expression using a try-except block.

If OptimizationOn property is true, Parse method will optimize the parse tree by evaluating constant branches of the parse tree at this moment, so that Evaluate function will run faster.

1.2.6.5 Evaluate

Unit
bcParser

Applies to
TbcParser

Declaration
function Evaluate : NUMBER ;virtual;

Description
Evaluate function returns the numerical value of the mathematical string Expression. If the expression was not parsed before, it will be parsed for the evaluation. Once the expression is parsed, it won't be parsed again and again to obtain the numeric value. Instead the output value will be efficiently calculated from the parse tree. If the Expression property is re-assigned a new mathematical expression, then expression will be parsed next time the Evaulate function or Parse method is called.

1.2.6.6 CreateVar

Unit
bcParser

Applies to
TbcParser

Declaration
procedure CreateVar (name : string ;value : NUMBER) ;virtual;

Description
CreateVar method creates a new variable in the parser's list of variables. If the variable name already exists, CreateVar throws EbcParserError exception.

1.2.6.7 CreateOneParamFunc

Unit
bcParser

Applies to
TbcParser

Declaration
procedure CreateOneParamFunc (name : string ;procAddr : TOneParamFunc) ;virtual;

Description
CreateOneParamFunc method creates a new function that takes one parameter in the parser's list of functions. If the function name already exists as a one parameter or two parameter function then CreateOneParamFunc throws EbcParserError exception.

1.2.6.8 CreateTwoParamFunc

Unit
bcParser

Applies to
TbcParser

Declaration
procedure CreateTwoParamFunc (name : string ;procAddr : TTwoParamFunc) ;virtual;

Description
CreateTwoParamFunc method creates a new function that takes two parameters in the parser's list of functions. If the function name already exists as a one parameter or two parameter function then CreateTwoParamFunc throws EbcParserError exception.

1.2.6.9 CreateNParamFunc

Unit
bcParser

Applies to
TbcParser

Declaration
procedure CreateNParamFunc (name : string ;procAddr : TNParamFunc ;nParam : Integer) ;virtual;

Description
CreateNParamFunc method creates a new function that takes n>2 parameters in the parser's list of functions. If the function name already exists as a function then CreateNParamFunc throws EbcParserError exception. nParam is the number of parameters which is also the size of the array that TNParamFunc takes as parameter.

1.2.6.10 CreateDefaultFuncs

Unit
bcParser

Applies to
TbcParser

Declaration
procedure CreateDefaultFuncs ;virtual;

Description
CreateDefaultFuncs method creates some predefined functions in the parser's list of functions.

Predefined functions that take one parameter are:

SQR: Square function which can be used as SQR(X)

SIN: Sinus function which can be used as SIN(X), X is a real-type expression. Sin returns the sine of the angle X in radians.

COS: Cosinus function which can be used as COS(X), X is a real-type expression. COS returns the

cosine of the angle X in radians.

ATAN: ArcTangent function which can be used as ATAN(X)

SINH: Sinus Hyperbolic function which can be used as SINH(X)

COSH: Cosinus Hyperbolic function which can be used as COSH(X)

COTAN: which can be used as COTAN(X)

TAN: which can be used as TAN(X)

EXP: which can be used as EXP(X)

LN: natural log, which can be used as LN(X)

LOG: 10 based log, which can be used as LOG(X)

SQRT: which can be used as SQRT(X)

ABS: absolute value, which can be used as ABS(X)

SIGN: SIGN(X) returns -1 if X<0; +1 if X>0, 0 if X=0; it can be used as SQR(X)

TRUNC: Discards the fractional part of a number. e.g. TRUNC(-3.2) is -3, TRUNC(3.2) is 3.

CEIL: CEIL(-3.2) = 3, CEIL(3.2) = 4

FLOOR: FLOOR(-3.2) = -4, FLOOR(3.2) = 3

RND: Random number generator.

RND(X) generates a random INTEGER number such that 0 <= Result < int(X). Call TbcParser.Randomize to initialize the random number generator with a random seed value before using RND function in your expression.

RANDOM: Random number generator.

RANDOM(X) generates a random floating point number such that 0 <= Result < X. Call TbcParser.Randomize to initialize the random number generator with a random seed value before using RANDOM function in your expression.

Predefined functions that take two parameters are:

INTPOW: The INTPOW function raises Base to an integral power. INTPOW(2, 3) = 8. Note that result of INTPOW(2, 3.4) = 8 as well.

POW: The Power function raises Base to any power. For fractional exponents or exponents greater than MaxInt, Base must be greater than 0.

LOGN: The LogN function returns the log base N of X. Example: LOGN(10, 100) = 2

MIN: MIN(2, 3) is 2.

MAX: MAX(2, 3) is 3.

Predefined functions that take more than 2 parameters are:

IF: IF(BOOL, X, Y) returns X if BOOL is <> 0, returns Y if BOOL =0. Values of X and Y are calculated regardless of BOOL (Full Boolean Evaluation).

1.2.6.11 CreateDefaultVars

Unit
bcParser

Applies to
TbcParser

Declaration
procedure CreateDefaultVars ;virtual;

Description
X, Y and PI variables are predefined and can be immediately used in the expression. Initial values of X and Y are 0. PI is 3.14159265358979

1.2.6.12 AssignVar

Unit
bcParser

Applies to
TbcParser

Declaration
procedure AssignVar (name : string ;val : NUMBER) ;virtual;

Description
AssignVar method assigns a given value 'val' to the variable named 'name'.

1.2.6.13 GetVar

Unit
bcParser

Applies to
TbcParser

Declaration

function GetVar (name : string) : NUMBER ;virtual;

Description
GetVar method returns the current value of the variable named 'name'.

1.2.6.14 DeleteVar

Unit
bcParser

Applies to
TbcParser

Declaration
procedure DeleteVar (name : string) ;virtual;

Description
DeleteVar method deletes an existing variable from the list of available variables. If the variable does not exist, then DeleteVar throws EbcParserError exception.

When a variable is deleted Dirty flag is set to true so that next time the Evaluate function is called the expression will be reparsed.

1.2.6.15 DeleteFunc

Unit
bcParser

Applies to
TbcParser

Declaration
procedure DeleteFunc (name : string) ;virtual;

Description
DeleteFunc method deletes an existing function from the list of available functions. If the function does not exist, then DeleteVar throws EbcParserError exception.

When a function is deleted Dirty flag is set to true so that next time the Evaluate function is called the expression will be reparsed.

1.2.6.16 DeleteAllVars

Unit
bcParser

Applies to
TbcParser

Declaration
procedure DeleteAllVars ;virtual;

Description
DeleteAllVars method deletes all variables from the list of available variables.

This action may be useful when number of unused variables is too high that causes performance to degrade.

When a variable is deleted Dirty flag is set to true so that next time the Evaluate function is called the expression will be reparsed.

1.2.6.17 DeleteAllFuncs

Unit
bcParser

Applies to
TbcParser

Declaration
procedure DeleteAllFuncs ;virtual;

Description
DeleteAllFuncs method deletes all variables from the list of available functions.

This action may be useful when number of unused functions is too high that causes performance to degrade.

When a function is deleted Dirty flag is set to true so that next time the Evaluate function is called the expression will be reparsed.

1.2.6.18 GetFunctions

Unit
bcParser

Applies to
TbcParser

Declaration
function GetFunctions : TStringList ;virtual;

Description

Returns a new TStringList object that contains all available function names. It is caller's responsibility to free the returned TStringList. Making changes to this list does not effect the parser.

1.2.6.19 GetVariables

Unit
bcParser

Applies to
TbcParser

Declaration
function GetVariables : TStringList ;virtual;

Description

Returns a new TStringList object that contains all available variable names. It is caller's responsibility to free the returned TStringList. Making changes to this list does not effect the parser.

1.2.6.20 GetVariablesUsed

Unit
bcParser

Applies to
TbcParser

Declaration
function GetVariablesUsed : TStringList ;virtual;

Description

Returns a new TStringList object that contains variable names that are used in the current expression. It is caller's responsibility to free the returned TStringList. Making changes to this list does not effect the parser.

1.2.6.21 Randomize

Unit
bcParser

Applies to
TbcParser

Declaration
procedure Randomize ;virtual;

Description
Randomize function should be called to initialize the predefined RND or RANDOM functions if the expression uses them.

1.2.6.22 FreeParseTree

Unit
bcParser

Applies to
TbcParser

Declaration
procedure FreeParseTree ;virtual;

Description
FreeParseTree can be explicitly called to free the resources taken by the allocated Parse tree when an expression is parsed. FreeParseTree sets the Dirty flag to true so that next time the Evaluate function is called, expression will be parsed forming a new, valid parse tree to be evaluated.

1.2.6.23 IsVariableUsed

Unit
bcParser

Applies to
TbcParser

Declaration
function IsVariableUsed (const varName : string) : boolean ;virtual;

Description
Returns true if a variable with the name 'varName' is used in the current expression.

1.2.6.24 IsFunctionUsed

Unit
bcParser

Applies to
TbcParser

Declaration
function IsFunctionUsed (const funcName : string) : boolean ;virtual;

Description
Returns true if a function with the name 'funcName' is used in the current expression.

1.2.6.25 Value

Unit
bcParser

Applies to
TbcParser

Declaration
property Value : NUMBER ;Read only

Description
Value property is an intuitive way of retrieving the value of the input expression. Value property is a read only property which is in fact just an alias for the Evaluate method.

1.2.6.26 Variable

Unit
bcParser

Applies to
TbcParser

Declaration
property Variable [name : string] : NUMBER ;

Description
Variable property is a way to set and get variable values.

1.2.6.27 IsVariable

Unit
bcParser

Applies to
TbcParser

Declaration
function IsVariable (const varName : string) : boolean ;

Description
some useful functions:
Returns true if a variable with the name 'varName' is present in the current variables list.

1.2.6.28 IsFunction

Unit
bcParser

Applies to
TbcParser

Declaration
function IsFunction (const funcName : string) : boolean ;

Description
Returns true if a function with the name 'funcName' is defined as a function.

1.2.6.29 IsOneParamFunction

Unit
bcParser

Applies to
TbcParser

Declaration
function IsOneParamFunction (const funcName : string) : boolean ;

Description
Returns true if a function with the name 'funcName' is present in the current one parameter functions list.

1.2.6.30 IsTwoParamFunction

Unit
bcParser

Applies to
TbcParser

Declaration
function IsTwoParamFunction (const funcName : string) : boolean ;

Description
Returns true if a function with the name 'funcName' is present in the current two parameter functions list.

1.2.6.31 IsNParamFunction

Unit
bcParser

Applies to
TbcParser

Declaration
function IsNParamFunction (const funcName : string) : boolean ;

Description
Returns true if a function with the name 'funcName' is present in the current list of functions that take any number of parameters.

1.2.6.32 Expression

Unit
bcParser

Applies to
TbcParser

Declaration
property Expression : string ;

Description

Expression property represents the mathematical expression which is input to be evaluated by the user.

The expression can contain variables such as X, Y, T, HEIGHT, WEIGHT and so on. Expression can also contain functions that take one parameter, two parameters or any number of parameters. Expressions are case insensitive. Internally they are converted to uppercase before parsing.

When Expression is assigned a value, it becomes 'dirty' and further attempt to evaluate its value will require it to be parsed. But once it is parsed, and a parse tree representing the expression is formed, it won't be parsed again, until it is assigned a new string. Instead, the parse tree will be used to retrieve current results as the values of variables change.

X, Y and PI variables are predefined and can be immediately used in the expression (If you don't want them, you can delete with DeleteVar). CreateVar method can be used to add user variables. Variables do not have to be predefined. You can choose to implement OnVariableFound event handler. If OnVariableFound event handler is assigned, then it will be invoked to retrieve values for undefined

variables.

Predefined functions that take one parameter are:

SQR: Square function which can be used as SQR(X)

SIN: Sinus function which can be used as SIN(X), X is a real-type expression. Sin returns the sine of the angle X in radians.

COS: Cosinus function which can be used as COS(X), X is a real-type expression. COS returns the cosine of the angle X in radians.

ATAN: ArcTangent function which can be used as ATAN(X)

SINH: Sinus Hyperbolic function which can be used as SINH(X)

COSH: Cosinus Hyperbolic function which can be used as COSH(X)

COTAN: which can be used as COTAN(X)

TAN: which can be used as TAN(X)

EXP: which can be used as EXP(X)

LN: natural log, which can be used as LN(X)

LOG: 10 based log, which can be used as LOG(X)

SQRT: which can be used as SQRT(X)

ABS: absolute value, which can be used as ABS(X)

SIGN: SIGN(X) returns -1 if X<0; +1 if X>0, 0 if X=0; it can be used as SQR(X)

TRUNC: Discards the fractional part of a number. e.g. TRUNC(-3.2) is -3, TRUNC(3.2) is 3.

CEIL: CEIL(-3.2) = 3, CEIL(3.2) = 4

FLOOR: FLOOR(-3.2) = -4, FLOOR(3.2) = 3

RND: Random number generator.

RND(X) generates a random INTEGER number such that 0 <= Result < int(X). Call TbcParser. Randomize to initialize the random number generator with a random seed value before using RND function in your expression.

RANDOM: Random number generator.

RANDOM(X) generates a random floating point number such that 0 <= Result < X. Call TbcParser. Randomize to initialize the random number generator with a random seed value before using RANDOM function in your expression.

Predefined functions that take two parameters are:

INTPOW: The INTPOW function raises Base to an integral power. INTPOW(2, 3) = 8. Note that result of INTPOW(2, 3.4) = 8 as well.

POW: The Power function raises Base to any power. For fractional exponents or exponents greater than MaxInt, Base must be greater than 0.

LOGN: The LogN function returns the log base N of X. Example: LOGN(10, 100) = 2

MIN: MIN(2, 3) is 2.

MAX: MAX(2, 3) is 3.

Predefined functions that take more than 2 parameters are:

IF: IF(BOOL, X, Y) returns X if BOOL is <> 0, returns Y if BOOL =0. Values of X and Y are calculated regardless of BOOL (Full Boolean Evaluation).

User functions can be added using CreateOneParamFunc, CreateTwoParamFunc and CreateNParamFunc methods. Functions and Variables can be deleted using DeleteVar, DeleteFunc, DeleteAllVars, DeleteAllFuncs methods.

1.2.6.33 X

Unit
bcParser

Applies to
TbcParser

Declaration
property X : NUMBER ;

Description

X property represents the X variable used in the mathematical expression which was input to be evaluated. You can set the X variable to a numeric value and call the Parse method (or Value property) to retrieve the new result of the expression. X variable is created by default for the convenience of the user. Additional variables can be added by using the CreateVar method. Variable names are case insensitive.

1.2.6.34 Y

Unit
bcParser

Applies to

TbcParser

Declaration
property Y : NUMBER ;

Description

Y property represents the Y variable used in the mathematical expression which was input to be evaluated. You can set the Y variable to a numeric value and call the Parse method (or Value property) to retrieve the new result of the expression. Y variable is created by default for the convenience of the user. Additional variables can be added by using the CreateVar method. Variable names are case insensitive.

1.2.6.35 OptimizationOn

Unit
bcParser

Applies to
TbcParser

Declaration
property OptimizationOn : boolean default true ;

Description

Set OptimizationOn to let the bcParser component evaluate constant expressions at parse time. The optimized parse tree will enhance subsequant evaluation operations, though initial parsing will be slower.

Optimization is good if you are going to parse once and evaluate the same expression many many times with different variable values.

When OptimizationOn is true, following code runs in 170 milliseconds. When Optimization is false, following code runs in 220 milliseconds on PII 300.

procedure TForm1.bbCalcClick(Sender: TObject); var i: integer; count: integer; begin bcParser. Expression := 'SIN(3.14)+5^2+POW(2,7)-MAX(10,20)';

count:= GetTickCount; try for i:= 1 to 10000 do begin bcParser.X := i*2; bcParser.Y := i/2; labResult. Caption := FloatToStr(bcParser.Parse); end; except on e: Exception do ShowMessage(e.Message); end;

ShowMessage(FloatToStr(GetTickCount - count)); end;

1.2.6.36 OnVariableFound

Unit
bcParser

Applies to
TbcParser

Declaration

procedure(Sender : TObject; const varName : String; var RetVal : NUMBER) of Object;

Description

OnVariableFound event handler is used to return the value of an undeclared variable on demand.

If OnVariableFound event handler is not assigned, TbcParser requires that variables are predefined prior to Parse( ) operation. However, if OnVariableFound event handler is assigned, Parser will tolerate undefined variables in the expression. After parse, when the expression's value is requested, TbcParser will invoke OnVariableFound event handler for each variable that was not previously assigned a value for.

When OnVariableFound is event handler is present, it is the user's responsibility to decide whether a variable is valid or not, and decide what the current value of the variable shall be.

OnVariableFound event is useful in cases when the application domain is too big to define all possible variables that can be used in a mathematical expression ahead of time.

Index

- B -

- C -

- E -

- I -

- N -

- O -

- T -

- V -

- X -

- Y -