

Tang

0.1

Generated by Doxygen 1.9.1

1 Tang: A Template Language	1
1.1 Quick Description	1
1.2 Features	1
1.3 License	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	11
5.1 Tang::AstNode Class Reference	11
5.1.1 Detailed Description	14
5.1.2 Constructor & Destructor Documentation	14
5.1.2.1 AstNode()	14
5.1.3 Member Function Documentation	14
5.1.3.1 compileIdentifiers()	14
5.1.3.2 makeCopy()	15
5.2 Tang::AstNodeAssign Class Reference	15
5.2.1 Detailed Description	17
5.2.2 Constructor & Destructor Documentation	17
5.2.2.1 AstNodeAssign()	17
5.2.3 Member Function Documentation	17
5.2.3.1 compileIdentifiers()	17
5.2.3.2 makeCopy()	18
5.3 Tang::AstNodeBinary Class Reference	18
5.3.1 Detailed Description	20
5.3.2 Member Enumeration Documentation	20
5.3.2.1 Operation	20
5.3.3 Constructor & Destructor Documentation	20
5.3.3.1 AstNodeBinary()	21
5.3.4 Member Function Documentation	21
5.3.4.1 compileIdentifiers()	21
5.3.4.2 makeCopy()	21
5.4 Tang::AstNodeBlock Class Reference	22
5.4.1 Detailed Description	24
5.4.2 Constructor & Destructor Documentation	24
5.4.2.1 AstNodeBlock()	24
5.4.3 Member Function Documentation	24
5.4.3.1 compileIdentifiers()	24

5.4.3.2 makeCopy()	25
5.5 Tang::AstNodeBoolean Class Reference	25
5.5.1 Detailed Description	27
5.5.2 Constructor & Destructor Documentation	27
5.5.2.1 AstNodeBoolean()	27
5.5.3 Member Function Documentation	27
5.5.3.1 compileIdentifiers()	27
5.5.3.2 makeCopy()	28
5.6 Tang::AstNodeCast Class Reference	28
5.6.1 Detailed Description	30
5.6.2 Member Enumeration Documentation	30
5.6.2.1 Type	30
5.6.3 Constructor & Destructor Documentation	30
5.6.3.1 AstNodeCast()	30
5.6.4 Member Function Documentation	31
5.6.4.1 compileIdentifiers()	31
5.6.4.2 makeCopy()	31
5.7 Tang::AstNodeFloat Class Reference	31
5.7.1 Detailed Description	33
5.7.2 Constructor & Destructor Documentation	33
5.7.2.1 AstNodeFloat()	33
5.7.3 Member Function Documentation	33
5.7.3.1 compileIdentifiers()	34
5.7.3.2 makeCopy()	34
5.8 Tang::AstNodeIdentifier Class Reference	34
5.8.1 Detailed Description	36
5.8.2 Constructor & Destructor Documentation	36
5.8.2.1 AstNodeIdentifier()	36
5.8.3 Member Function Documentation	36
5.8.3.1 compileIdentifiers()	36
5.8.3.2 makeCopy()	37
5.9 Tang::AstNodeInteger Class Reference	37
5.9.1 Detailed Description	39
5.9.2 Constructor & Destructor Documentation	39
5.9.2.1 AstNodeInteger()	39
5.9.3 Member Function Documentation	39
5.9.3.1 compileIdentifiers()	39
5.9.3.2 makeCopy()	40
5.10 Tang::AstNodeNull Class Reference	40
5.10.1 Detailed Description	42
5.10.2 Constructor & Destructor Documentation	42
5.10.2.1 AstNodeNull()	42

5.10.3 Member Function Documentation	42
5.10.3.1 compileIdentifiers()	42
5.10.3.2 makeCopy()	42
5.11 Tang::AstNodeUnary Class Reference	43
5.11.1 Detailed Description	45
5.11.2 Member Enumeration Documentation	45
5.11.2.1 Operator	45
5.11.3 Constructor & Destructor Documentation	45
5.11.3.1 AstNodeUnary()	45
5.11.4 Member Function Documentation	46
5.11.4.1 compileIdentifiers()	46
5.11.4.2 makeCopy()	46
5.12 Tang::ComputedExpression Class Reference	46
5.12.1 Detailed Description	48
5.12.2 Member Function Documentation	48
5.12.2.1 __add()	48
5.12.2.2 __boolean()	48
5.12.2.3 __divide()	49
5.12.2.4 __equal()	49
5.12.2.5 __float()	50
5.12.2.6 __integer()	50
5.12.2.7 __lessThan()	50
5.12.2.8 __modulo()	51
5.12.2.9 __multiply()	51
5.12.2.10 __negative()	51
5.12.2.11 __not()	52
5.12.2.12 __subtract()	52
5.12.2.13 dump()	52
5.12.2.14 is_equal() [1/5]	52
5.12.2.15 is_equal() [2/5]	53
5.12.2.16 is_equal() [3/5]	53
5.12.2.17 is_equal() [4/5]	54
5.12.2.18 is_equal() [5/5]	54
5.12.2.19 makeCopy()	54
5.13 Tang::ComputedExpressionBoolean Class Reference	55
5.13.1 Detailed Description	56
5.13.2 Constructor & Destructor Documentation	56
5.13.2.1 ComputedExpressionBoolean()	56
5.13.3 Member Function Documentation	57
5.13.3.1 __add()	57
5.13.3.2 __boolean()	57
5.13.3.3 __divide()	57

5.13.3.4 <code>__equal()</code>	58
5.13.3.5 <code>__float()</code>	58
5.13.3.6 <code>__integer()</code>	59
5.13.3.7 <code>__lessThan()</code>	59
5.13.3.8 <code>__modulo()</code>	59
5.13.3.9 <code>__multiply()</code>	60
5.13.3.10 <code>__negative()</code>	60
5.13.3.11 <code>__not()</code>	60
5.13.3.12 <code>__subtract()</code>	60
5.13.3.13 <code>dump()</code>	61
5.13.3.14 <code>is_equal()</code> [1/5]	61
5.13.3.15 <code>is_equal()</code> [2/5]	61
5.13.3.16 <code>is_equal()</code> [3/5]	62
5.13.3.17 <code>is_equal()</code> [4/5]	62
5.13.3.18 <code>is_equal()</code> [5/5]	63
5.13.3.19 <code>makeCopy()</code>	63
5.14 Tang::ComputedExpressionError Class Reference	64
5.14.1 Detailed Description	65
5.14.2 Constructor & Destructor Documentation	65
5.14.2.1 <code>ComputedExpressionError()</code>	65
5.14.3 Member Function Documentation	66
5.14.3.1 <code>__add()</code>	66
5.14.3.2 <code>__boolean()</code>	66
5.14.3.3 <code>__divide()</code>	66
5.14.3.4 <code>__equal()</code>	67
5.14.3.5 <code>__float()</code>	67
5.14.3.6 <code>__integer()</code>	68
5.14.3.7 <code>__lessThan()</code>	68
5.14.3.8 <code>__modulo()</code>	68
5.14.3.9 <code>__multiply()</code>	69
5.14.3.10 <code>__negative()</code>	69
5.14.3.11 <code>__not()</code>	69
5.14.3.12 <code>__subtract()</code>	69
5.14.3.13 <code>dump()</code>	70
5.14.3.14 <code>is_equal()</code> [1/5]	70
5.14.3.15 <code>is_equal()</code> [2/5]	70
5.14.3.16 <code>is_equal()</code> [3/5]	71
5.14.3.17 <code>is_equal()</code> [4/5]	71
5.14.3.18 <code>is_equal()</code> [5/5]	72
5.14.3.19 <code>makeCopy()</code>	72
5.15 Tang::ComputedExpressionFloat Class Reference	73
5.15.1 Detailed Description	74

5.15.2 Constructor & Destructor Documentation	74
5.15.2.1 ComputedExpressionFloat()	74
5.15.3 Member Function Documentation	75
5.15.3.1 __add()	75
5.15.3.2 __boolean()	75
5.15.3.3 __divide()	75
5.15.3.4 __equal()	76
5.15.3.5 __float()	76
5.15.3.6 __integer()	77
5.15.3.7 __lessThan()	77
5.15.3.8 __modulo()	77
5.15.3.9 __multiply()	78
5.15.3.10 __negative()	78
5.15.3.11 __not()	78
5.15.3.12 __subtract()	78
5.15.3.13 dump()	79
5.15.3.14 is_equal() [1/5]	79
5.15.3.15 is_equal() [2/5]	79
5.15.3.16 is_equal() [3/5]	80
5.15.3.17 is_equal() [4/5]	80
5.15.3.18 is_equal() [5/5]	81
5.15.3.19 makeCopy()	81
5.16 Tang::ComputedExpressionInteger Class Reference	82
5.16.1 Detailed Description	83
5.16.2 Constructor & Destructor Documentation	83
5.16.2.1 ComputedExpressionInteger()	83
5.16.3 Member Function Documentation	84
5.16.3.1 __add()	84
5.16.3.2 __boolean()	84
5.16.3.3 __divide()	84
5.16.3.4 __equal()	85
5.16.3.5 __float()	85
5.16.3.6 __integer()	86
5.16.3.7 __lessThan()	86
5.16.3.8 __modulo()	86
5.16.3.9 __multiply()	87
5.16.3.10 __negative()	87
5.16.3.11 __not()	87
5.16.3.12 __subtract()	87
5.16.3.13 dump()	88
5.16.3.14 is_equal() [1/5]	88
5.16.3.15 is_equal() [2/5]	88

5.16.3.16 is_equal() [3/5]	89
5.16.3.17 is_equal() [4/5]	89
5.16.3.18 is_equal() [5/5]	90
5.16.3.19 makeCopy()	90
5.17 Tang::ComputedExpressionNull Class Reference	91
5.17.1 Detailed Description	92
5.17.2 Member Function Documentation	92
5.17.2.1 __add()	92
5.17.2.2 __boolean()	93
5.17.2.3 __divide()	93
5.17.2.4 __equal()	93
5.17.2.5 __float()	94
5.17.2.6 __integer()	94
5.17.2.7 __lessThan()	94
5.17.2.8 __modulo()	95
5.17.2.9 __multiply()	95
5.17.2.10 __negative()	96
5.17.2.11 __not()	96
5.17.2.12 __subtract()	96
5.17.2.13 dump()	97
5.17.2.14 is_equal() [1/5]	97
5.17.2.15 is_equal() [2/5]	97
5.17.2.16 is_equal() [3/5]	98
5.17.2.17 is_equal() [4/5]	98
5.17.2.18 is_equal() [5/5]	98
5.17.2.19 makeCopy()	99
5.18 Tang::Error Class Reference	99
5.18.1 Detailed Description	101
5.18.2 Constructor & Destructor Documentation	101
5.18.2.1 Error() [1/2]	101
5.18.2.2 Error() [2/2]	101
5.18.3 Friends And Related Function Documentation	101
5.18.3.1 operator<<	102
5.19 Tang::GarbageCollected Class Reference	102
5.19.1 Detailed Description	104
5.19.2 Constructor & Destructor Documentation	104
5.19.2.1 GarbageCollected() [1/3]	104
5.19.2.2 GarbageCollected() [2/3]	105
5.19.2.3 ~GarbageCollected()	105
5.19.2.4 GarbageCollected() [3/3]	105
5.19.3 Member Function Documentation	105
5.19.3.1 make()	105

5.19.3.2 operator"!()	106
5.19.3.3 operator"!=()	106
5.19.3.4 operator%()	107
5.19.3.5 operator*() [1/2]	108
5.19.3.6 operator*() [2/2]	108
5.19.3.7 operator+()	108
5.19.3.8 operator-() [1/2]	109
5.19.3.9 operator-() [2/2]	109
5.19.3.10 operator->()	110
5.19.3.11 operator/()	110
5.19.3.12 operator<()	111
5.19.3.13 operator<=()	111
5.19.3.14 operator=() [1/2]	112
5.19.3.15 operator=() [2/2]	112
5.19.3.16 operator==() [1/6]	113
5.19.3.17 operator==() [2/6]	113
5.19.3.18 operator==() [3/6]	114
5.19.3.19 operator==() [4/6]	114
5.19.3.20 operator==() [5/6]	114
5.19.3.21 operator==() [6/6]	115
5.19.3.22 operator>()	115
5.19.3.23 operator>=()	115
5.19.4 Friends And Related Function Documentation	117
5.19.4.1 operator<<	117
5.20 Tang::location Class Reference	118
5.20.1 Detailed Description	119
5.21 Tang::position Class Reference	119
5.21.1 Detailed Description	120
5.22 Tang::Program Class Reference	121
5.22.1 Detailed Description	122
5.22.2 Member Enumeration Documentation	122
5.22.2.1 CodeType	122
5.22.3 Constructor & Destructor Documentation	123
5.22.3.1 Program()	123
5.22.4 Member Function Documentation	123
5.22.4.1 addBytecode()	123
5.22.4.2 dumpBytecode()	123
5.22.4.3 execute()	124
5.22.4.4 getAst()	124
5.22.4.5 getCode()	124
5.22.4.6 getResult()	124
5.23 Tang::SingletonObjectPool< T > Class Template Reference	125

5.23.1 Detailed Description	125
5.23.2 Member Function Documentation	125
5.23.2.1 get()	125
5.23.2.2 getInstance()	126
5.23.2.3 recycle()	126
5.24 Tang::TangBase Class Reference	126
5.24.1 Detailed Description	127
5.24.2 Constructor & Destructor Documentation	127
5.24.2.1 TangBase()	127
5.24.3 Member Function Documentation	127
5.24.3.1 compileScript()	127
5.25 Tang::TangScanner Class Reference	128
5.25.1 Detailed Description	128
5.25.2 Constructor & Destructor Documentation	129
5.25.2.1 TangScanner()	129
5.25.3 Member Function Documentation	129
5.25.3.1 get_next_token()	129
6 File Documentation	131
6.1 build/generated/location.hh File Reference	131
6.1.1 Detailed Description	132
6.1.2 Function Documentation	132
6.1.2.1 operator<<() [1/2]	132
6.1.2.2 operator<<() [2/2]	133
6.2 include/astNode.hpp File Reference	133
6.2.1 Detailed Description	134
6.3 include/astNodeAssign.hpp File Reference	134
6.3.1 Detailed Description	135
6.4 include/astNodeBinary.hpp File Reference	135
6.4.1 Detailed Description	136
6.5 include/astNodeBlock.hpp File Reference	136
6.5.1 Detailed Description	137
6.6 include/astNodeBoolean.hpp File Reference	137
6.6.1 Detailed Description	138
6.7 include/astNodeCast.hpp File Reference	138
6.7.1 Detailed Description	139
6.8 include/astNodeFloat.hpp File Reference	139
6.8.1 Detailed Description	140
6.9 include/astNodeIdentifier.hpp File Reference	140
6.9.1 Detailed Description	141
6.10 include/astNodeInteger.hpp File Reference	141
6.10.1 Detailed Description	142

6.11 include/astNodeNull.hpp File Reference	142
6.11.1 Detailed Description	143
6.12 include/astNodeUnary.hpp File Reference	143
6.12.1 Detailed Description	144
6.13 include/computedExpression.hpp File Reference	144
6.13.1 Detailed Description	145
6.14 include/computedExpressionBoolean.hpp File Reference	145
6.14.1 Detailed Description	146
6.15 include/computedExpressionError.hpp File Reference	146
6.15.1 Detailed Description	147
6.16 include/computedExpressionFloat.hpp File Reference	147
6.16.1 Detailed Description	147
6.17 include/computedExpressionInteger.hpp File Reference	148
6.17.1 Detailed Description	148
6.18 include/computedExpressionNull.hpp File Reference	149
6.18.1 Detailed Description	149
6.19 include/error.hpp File Reference	150
6.19.1 Detailed Description	150
6.20 include/garbageCollected.hpp File Reference	151
6.20.1 Detailed Description	151
6.21 include/macros.hpp File Reference	151
6.21.1 Detailed Description	152
6.21.2 Macro Definition Documentation	152
6.21.2.1 TANG_UNUSED	152
6.22 include/opcode.hpp File Reference	152
6.22.1 Detailed Description	153
6.22.2 Enumeration Type Documentation	153
6.22.2.1 Opcode	153
6.23 include/program.hpp File Reference	153
6.23.1 Detailed Description	154
6.24 include/singletonObjectPool.hpp File Reference	155
6.24.1 Detailed Description	155
6.25 include/tang.hpp File Reference	156
6.25.1 Detailed Description	156
6.26 include/tangBase.hpp File Reference	157
6.26.1 Detailed Description	158
6.27 include/tangScanner.hpp File Reference	158
6.27.1 Detailed Description	159
6.28 src/astNode.cpp File Reference	159
6.28.1 Detailed Description	159
6.29 src/astNodeAssign.cpp File Reference	159
6.29.1 Detailed Description	160

6.30 src/astNodeBinary.cpp File Reference	160
6.30.1 Detailed Description	161
6.31 src/astNodeBlock.cpp File Reference	161
6.31.1 Detailed Description	161
6.32 src/astNodeBoolean.cpp File Reference	161
6.32.1 Detailed Description	162
6.33 src/astNodeCast.cpp File Reference	162
6.33.1 Detailed Description	163
6.34 src/astNodeFloat.cpp File Reference	163
6.34.1 Detailed Description	164
6.35 src/astNodeIdentifier.cpp File Reference	164
6.35.1 Detailed Description	165
6.36 src/astNodeInteger.cpp File Reference	165
6.36.1 Detailed Description	166
6.37 src/astNodeNull.cpp File Reference	166
6.37.1 Detailed Description	167
6.38 src/astNodeUnary.cpp File Reference	167
6.38.1 Detailed Description	168
6.39 src/computedExpression.cpp File Reference	168
6.39.1 Detailed Description	169
6.40 src/computedExpressionBoolean.cpp File Reference	169
6.40.1 Detailed Description	170
6.41 src/computedExpressionError.cpp File Reference	170
6.41.1 Detailed Description	170
6.42 src/computedExpressionFloat.cpp File Reference	170
6.42.1 Detailed Description	171
6.43 src/computedExpressionInteger.cpp File Reference	171
6.43.1 Detailed Description	171
6.44 src/computedExpressionNull.cpp File Reference	172
6.44.1 Detailed Description	172
6.45 src/error.cpp File Reference	172
6.45.1 Detailed Description	173
6.45.2 Function Documentation	173
6.45.2.1 operator<<()	173
6.46 src/program-dumpBytecode.cpp File Reference	174
6.46.1 Detailed Description	174
6.46.2 Macro Definition Documentation	174
6.46.2.1 DUMPPROGRAMCHECK	175
6.47 src/program-execute.cpp File Reference	175
6.47.1 Detailed Description	176
6.47.2 Macro Definition Documentation	176
6.47.2.1 EXECUTEPROGRAMCHECK	176

6.47.2.2 STACKCHECK	176
6.48 src/program.cpp File Reference	176
6.48.1 Detailed Description	177
6.49 src/tangBase.cpp File Reference	177
6.49.1 Detailed Description	178
6.50 test/test.cpp File Reference	178
6.50.1 Detailed Description	179
6.51 test/testGarbageCollected.cpp File Reference	179
6.51.1 Detailed Description	179
6.52 test/testSingletonObjectPool.cpp File Reference	180
6.52.1 Detailed Description	180
Index	181

Chapter 1

Tang: A Template Language

1.1 Quick Description

Tang is a C++ Template Language. It takes the form of a library which may be included in other projects. It is under active development, and you can follow its progress here:

- [YouTube playlist](#)
- [GitHub repository](#)

1.2 Features

The following features are planned:

- Native support for Unicode/Utf-8 strings.
- Change from template to script mode using escape tags like PHP.
- Loosely typed, with Python-like indexing and slicing of containers.
- Syntax similar to C/C++/PHP.
- Code compiles to a custom Bytecode and is executed by the Tang VM.
- Fast and thread-safe.

1.3 License

MIT License

Copyright (c) 2022 Corey Pennycuff

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Tang::AstNode	11
Tang::AstNodeAssign	15
Tang::AstNodeBinary	18
Tang::AstNodeBlock	22
Tang::AstNodeBoolean	25
Tang::AstNodeCast	28
Tang::AstNodeFloat	31
Tang::AstNodeIdentifier	34
Tang::AstNodeInteger	37
Tang::AstNodeNull	40
Tang::AstNodeUnary	43
Tang::ComputedExpression	46
Tang::ComputedExpressionBoolean	55
Tang::ComputedExpressionError	64
Tang::ComputedExpressionFloat	73
Tang::ComputedExpressionInteger	82
Tang::ComputedExpressionNull	91
Tang::Error	99
Tang::GarbageCollected	102
Tang::location	118
Tang::position	119
Tang::Program	121
Tang::SingletonObjectPool< T >	125
Tang::TangBase	126
TangTangFlexLexer	
Tang::TangScanner	128

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Tang::AstNode	Base class for representing nodes of an Abstract Syntax Tree (AST)	11
Tang::AstNodeAssign	An AstNode that represents a binary expression	15
Tang::AstNodeBinary	An AstNode that represents a binary expression	18
Tang::AstNodeBlock	An AstNode that represents a code block	22
Tang::AstNodeBoolean	An AstNode that represents a boolean literal	25
Tang::AstNodeCast	An AstNode that represents a typecast of an expression	28
Tang::AstNodeFloat	An AstNode that represents an float literal	31
Tang::AstNodeIdentifier	An AstNode that represents an identifier	34
Tang::AstNodeInteger	An AstNode that represents an integer literal	37
Tang::AstNodeNull	An AstNode that represents a NULL value	40
Tang::AstNodeUnary	An AstNode that represents a unary negation	43
Tang::ComputedExpression	Represents the result of a computation that has been executed	46
Tang::ComputedExpressionBoolean	Represents an Boolean that is the result of a computation	55
Tang::ComputedExpressionError	Represents a Runtime Error	64
Tang::ComputedExpressionFloat	Represents a Float that is the result of a computation	73
Tang::ComputedExpressionInteger	Represents an Integer that is the result of a computation	82
Tang::ComputedExpressionNull	Represents an Null that is the result of a computation	91

Tang::Error	
Used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error	99
Tang::GarbageCollected	
A container that acts as a resource-counting garbage collector for the specified type	102
Tang::location	
Two points in a source file	118
Tang::position	
A point in a source file	119
Tang::Program	
Represents a compiled script or template that may be executed	121
Tang::SingletonObjectPool< T >	
A thread-safe, singleton object pool of the designated type	125
Tang::TangBase	
The base class for the Tang programming language	126
Tang::TangScanner	
The Flex lexer class for the main Tang language	128

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

build/generated/location.hh	
Define the <code>Tang::location</code> class	131
include/astNode.hpp	
Declare the <code>Tang::AstNode</code> base class	133
include/astNodeAssign.hpp	
Declare the <code>Tang::AstNodeAssign</code> class	134
include/astNodeBinary.hpp	
Declare the <code>Tang::AstNodeBinary</code> class	135
include/astNodeBlock.hpp	
Declare the <code>Tang::AstNodeBlock</code> class	136
include/astNodeBoolean.hpp	
Declare the <code>Tang::AstNodeBoolean</code> class	137
include/astNodeCast.hpp	
Declare the <code>Tang::AstNodeCast</code> class	138
include/astNodeFloat.hpp	
Declare the <code>Tang::AstNodeFloat</code> class	139
include/astNodeIdentifier.hpp	
Declare the <code>Tang::AstNodeIdentifier</code> class	140
include/astNodeInteger.hpp	
Declare the <code>Tang::AstNodeInteger</code> class	141
include/astNodeNull.hpp	
Declare the <code>Tang::AstNodeNull</code> class	142
include/astNodeUnary.hpp	
Declare the <code>Tang::AstNodeUnary</code> class	143
include/computedExpression.hpp	
Declare the <code>Tang::ComputedExpression</code> base class	144
include/computedExpressionBoolean.hpp	
Declare the <code>Tang::ComputedExpressionBoolean</code> class	145
include/computedExpressionError.hpp	
Declare the <code>Tang::ComputedExpressionError</code> class	146
include/computedExpressionFloat.hpp	
Declare the <code>Tang::ComputedExpressionFloat</code> class	147
include/computedExpressionInteger.hpp	
Declare the <code>Tang::ComputedExpressionInteger</code> class	148
include/computedExpressionNull.hpp	
Declare the <code>Tang::ComputedExpressionNull</code> class	149

include/error.hpp	150
Declare the Tang::Error class used to describe syntax and runtime errors	
include/garbageCollected.hpp	151
Declare the Tang::GarbageCollected class	
include/macros.hpp	151
Contains generic macros	
include/opcode.hpp	152
Declare the Opcodes used in the Bytecode representation of a program	
include/program.hpp	153
Declare the Tang::Program class used to compile and execute source code	
include/singletonObjectPool.hpp	155
Declare the Tang::SingletonObjectPool class	
include/tang.hpp	156
Header file supplied for use by 3rd party code so that they can easily include all necessary headers	
include/tangBase.hpp	157
Declare the Tang::TangBase class used to interact with Tang	
include/tangScanner.hpp	158
Declare the Tang::TangScanner used to tokenize a Tang script	
src/astNode.cpp	159
Define the Tang::AstNode class	
src/astNodeAssign.cpp	159
Define the Tang::AstNodeAssign class	
src/astNodeBinary.cpp	160
Define the Tang::AstNodeBinary class	
src/astNodeBlock.cpp	161
Define the Tang::AstNodeBlock class	
src/astNodeBoolean.cpp	161
Define the Tang::AstNodeBoolean class	
src/astNodeCast.cpp	162
Define the Tang::AstNodeCast class	
src/astNodeFloat.cpp	163
Define the Tang::AstNodeFloat class	
src/astNodeIdentifier.cpp	164
Define the Tang::AstNodeIdentifier class	
src/astNodeInteger.cpp	165
Define the Tang::AstNodeInteger class	
src/astNodeNull.cpp	166
Define the Tang::AstNodeNull class	
src/astNodeUnary.cpp	167
Define the Tang::AstNodeUnary class	
src/computedExpression.cpp	168
Define the Tang::ComputedExpression class	
src/computedExpressionBoolean.cpp	169
Define the Tang::ComputedExpressionBoolean class	
src/computedExpressionError.cpp	170
Define the Tang::ComputedExpressionError class	
src/computedExpressionFloat.cpp	170
Define the Tang::ComputedExpressionFloat class	
src/computedExpressionInteger.cpp	171
Define the Tang::ComputedExpressionInteger class	
src/computedExpressionNull.cpp	172
Define the Tang::ComputedExpressionNull class	
src/error.cpp	172
Define the Tang::Error class	
src/program-dumpBytecode.cpp	174
Define the Tang::Program::dumpBytecode method	

src/ program-execute.cpp	
Define the Tang::Program::execute method	175
src/ program.cpp	
Define the Tang::Program class	176
src/ tangBase.cpp	
Define the Tang::TangBase class	177
test/ test.cpp	
Test the general language behaviors	178
test/ testGarbageCollected.cpp	
Test the generic behavior of the Tang::GarbageCollected class	179
test/ testSingletonObjectPool.cpp	
Test the generic behavior of the Tang::SingletonObjectPool class	180

Chapter 5

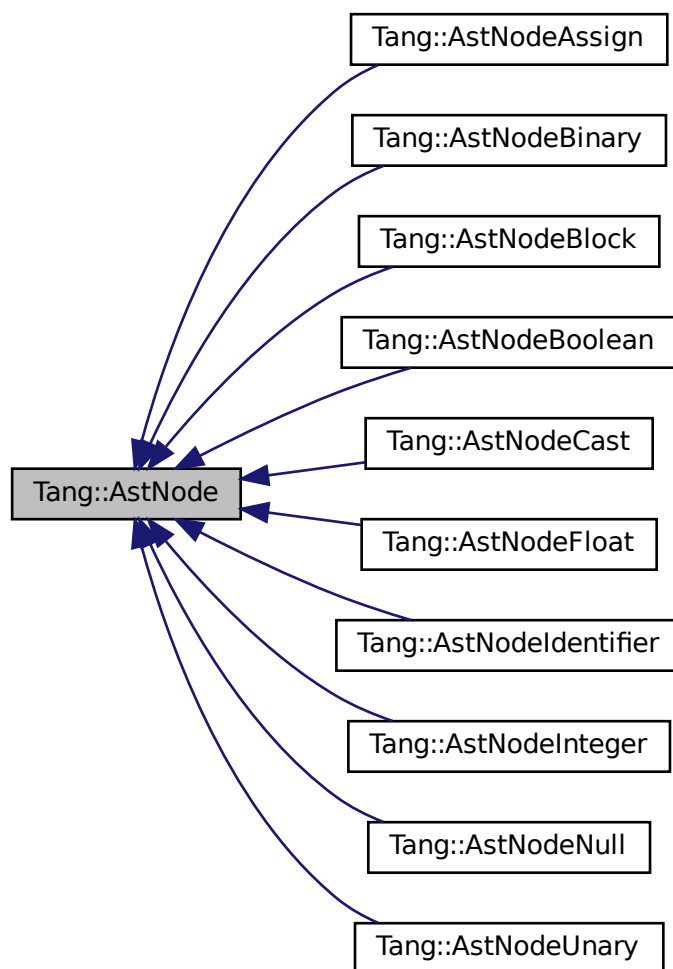
Class Documentation

5.1 Tang::AstNode Class Reference

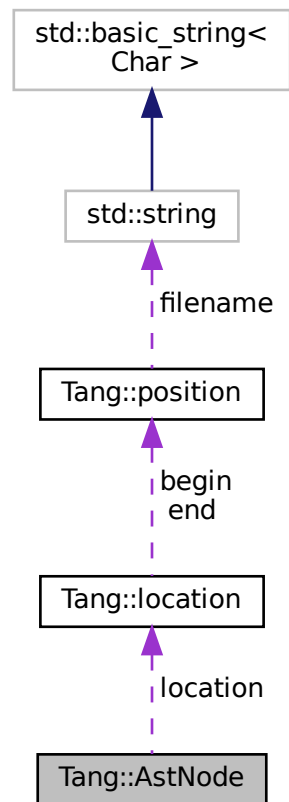
Base class for representing nodes of an Abstract Syntax Tree (AST).

```
#include <astNode.hpp>
```

Inheritance diagram for Tang::AstNode:



Collaboration diagram for Tang::AstNode:



Public Member Functions

- virtual `~AstNode` ()
The object destructor.
- virtual `std::string dump` (std::string indent="") const
Return a string that describes the contents of the node.
- virtual void `compile` (Tang::Program &program) const
Compile the ast of the provided Tang::Program.
- virtual `std::shared_ptr< AstNode > makeCopy` () const
Provide a copy of the AstNode (recursively, if appropriate).
- virtual void `compileIdentifiers` (Program &program) const
Compile a list of all variables in the scope.

Protected Member Functions

- `AstNode` (Tang::location location)
The generic constructor.

Protected Attributes

- [Tang::location location](#)

The location associated with this node.

5.1.1 Detailed Description

Base class for representing nodes of an Abstract Syntax Tree (AST).

There will be *many* derived classes, each one conveying the syntactic meaning of the code that it represents.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AstNode()

```
AstNode::AstNode (
    Tang::location location ) [protected]
```

The generic constructor.

It should never be called on its own.

Parameters

<i>location</i>	The location associated with this node.
-----------------	---

5.1.3 Member Function Documentation

5.1.3.1 compileIdentifiers()

```
void AstNode::compileIdentifiers (
    Program & program ) const [virtual]
```

Compile a list of all variables in the scope.

Parameters

<i>program</i>	The Tang::Program that is being compiled.
----------------	---

Reimplemented in [Tang::AstNodeIdentifier](#), [Tang::AstNodeBlock](#), [Tang::AstNodeBinary](#), and [Tang::AstNodeAssign](#).

5.1.3.2 makeCopy()

```
shared_ptr< AstNode > AstNode::makeCopy ( ) const [virtual]
```

Provide a copy of the [AstNode](#) (recursively, if appropriate).

Returns

A pointer to a new [AstNode](#) that is a copy of the current [AstNode](#).

Reimplemented in [Tang::AstNodeUnary](#), [Tang::AstNodeNull](#), [Tang::AstNodeInteger](#), [Tang::AstNodeIdentifier](#), [Tang::AstNodeFloat](#), [Tang::AstNodeCast](#), [Tang::AstNodeBoolean](#), [Tang::AstNodeBlock](#), [Tang::AstNodeBinary](#), and [Tang::AstNodeAssign](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

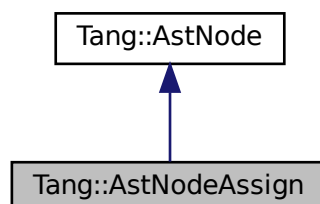
- [include/astNode.hpp](#)
- [src/astNode.cpp](#)

5.2 Tang::AstNodeAssign Class Reference

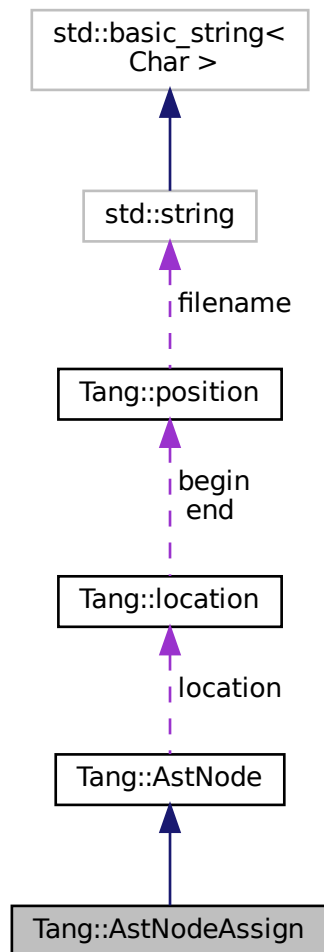
An [AstNode](#) that represents a binary expression.

```
#include <astNodeAssign.hpp>
```

Inheritance diagram for `Tang::AstNodeAssign`:



Collaboration diagram for Tang::AstNodeAssign:



Public Member Functions

- [AstNodeAssign](#) (`std::shared_ptr< AstNode > lhs`, `std::shared_ptr< AstNode > rhs`, [Tang::location](#) `location`)
The constructor.
- virtual `std::string` [dump](#) (`std::string indent=""`) const override
Return a string that describes the contents of the node.
- virtual void [compile](#) ([Tang::Program](#) &program) const override
Compile the ast of the provided [Tang::Program](#).
- virtual `shared_ptr< AstNode >` [makeCopy](#) () const override
Provide a copy of the [AstNode](#) (recursively, if appropriate).
- virtual void [compileIdentifiers](#) ([Program](#) &program) const override
Compile a list of all variables in the scope.

Protected Attributes

- [Tang::location](#) `location`

The location associated with this node.

5.2.1 Detailed Description

An [AstNode](#) that represents a binary expression.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AstNodeAssign()

```
AstNodeAssign::AstNodeAssign (
    std::shared_ptr< AstNode > lhs,
    std::shared_ptr< AstNode > rhs,
    Tang::location location )
```

The constructor.

Parameters

<i>lhs</i>	The left hand side expression.
<i>rhs</i>	The right hand side expression.
<i>location</i>	The location associated with the expression.

5.2.3 Member Function Documentation

5.2.3.1 compileIdentifiers()

```
void AstNodeAssign::compileIdentifiers (
    Program & program ) const [override], [virtual]
```

Compile a list of all variables in the scope.

Parameters

<i>program</i>	The Tang::Program that is being compiled.
----------------	---

Reimplemented from [Tang::AstNode](#).

5.2.3.2 makeCopy()

```
shared_ptr< AstNode > AstNodeAssign::makeCopy ( ) const [override], [virtual]
```

Provide a copy of the [AstNode](#) (recursively, if appropriate).

Returns

A pointer to a new [AstNode](#) that is a copy of the current [AstNode](#).

Reimplemented from [Tang::AstNode](#).

The documentation for this class was generated from the following files:

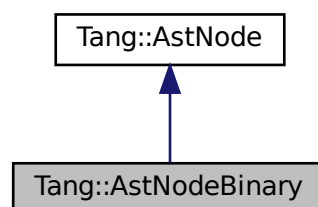
- include/[astNodeAssign.hpp](#)
- src/[astNodeAssign.cpp](#)

5.3 Tang::AstNodeBinary Class Reference

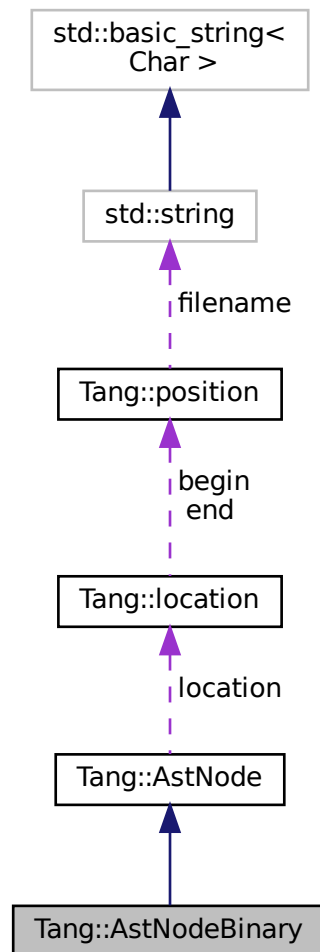
An [AstNode](#) that represents a binary expression.

```
#include <astNodeBinary.hpp>
```

Inheritance diagram for Tang::AstNodeBinary:



Collaboration diagram for Tang::AstNodeBinary:



Public Types

- enum [Operation](#) {
[Add](#) , [Subtract](#) , [Multiply](#) , [Divide](#) ,
[Modulo](#) , [LessThan](#) , [LessThanEqual](#) , [GreaterThan](#) ,
[GreaterThanEqual](#) , [Equal](#) , [NotEqual](#) }

Public Member Functions

- [AstNodeBinary](#) ([Operation](#) op, std::shared_ptr< [AstNode](#) > lhs, std::shared_ptr< [AstNode](#) > rhs, [Tang::location](#) location)
The constructor.
- virtual std::string [dump](#) (std::string indent="") const override
Return a string that describes the contents of the node.

- virtual void `compile` (`Tang::Program` &program) const override
Compile the ast of the provided `Tang::Program`.
- virtual shared_ptr< `AstNode` > `makeCopy` () const override
Provide a copy of the `AstNode` (recursively, if appropriate).
- virtual void `compileIdentifiers` (`Program` &program) const override
Compile a list of all variables in the scope.

Protected Attributes

- `Tang::location` location
The location associated with this node.

5.3.1 Detailed Description

An `AstNode` that represents a binary expression.

5.3.2 Member Enumeration Documentation

5.3.2.1 Operation

```
enum Tang::AstNodeBinary::Operation
```

Enumerator

Add	Indicates lhs + rhs.
Subtract	Indicates lhs - rhs.
Multiply	Indicates lhs * rhs.
Divide	Indicates lhs / rhs.
Modulo	Indicates lhs % rhs.
LessThan	Indicates lhs < rhs.
LessThanEqual	Indicates lhs <= rhs.
GreaterThan	Indicates lhs > rhs.
GreaterThanEqual	Indicates lhs >= rhs.
Equal	Indicates lhs == rhs.
NotEqual	Indicates lhs != rhs.

5.3.3 Constructor & Destructor Documentation

5.3.3.1 AstNodeBinary()

```
AstNodeBinary::AstNodeBinary (
    Operation op,
    std::shared_ptr< AstNode > lhs,
    std::shared_ptr< AstNode > rhs,
    Tang::location location )
```

The constructor.

Parameters

<i>op</i>	The Tang::AstNodeBinary::Operation to perform.
<i>lhs</i>	The left hand side expression.
<i>rhs</i>	The right hand side expression.
<i>location</i>	The location associated with the expression.

5.3.4 Member Function Documentation

5.3.4.1 compileIdentifiers()

```
void AstNodeBinary::compileIdentifiers (
    Program & program ) const [override], [virtual]
```

Compile a list of all variables in the scope.

Parameters

<i>program</i>	The Tang::Program that is being compiled.
----------------	---

Reimplemented from [Tang::AstNode](#).

5.3.4.2 makeCopy()

```
shared_ptr< AstNode > AstNodeBinary::makeCopy ( ) const [override], [virtual]
```

Provide a copy of the [AstNode](#) (recursively, if appropriate).

Returns

A pointer to a new [AstNode](#) that is a copy of the current [AstNode](#).

Reimplemented from [Tang::AstNode](#).

The documentation for this class was generated from the following files:

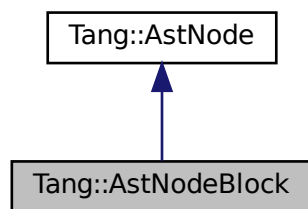
- include/[astNodeBinary.hpp](#)
- src/[astNodeBinary.cpp](#)

5.4 Tang::AstNodeBlock Class Reference

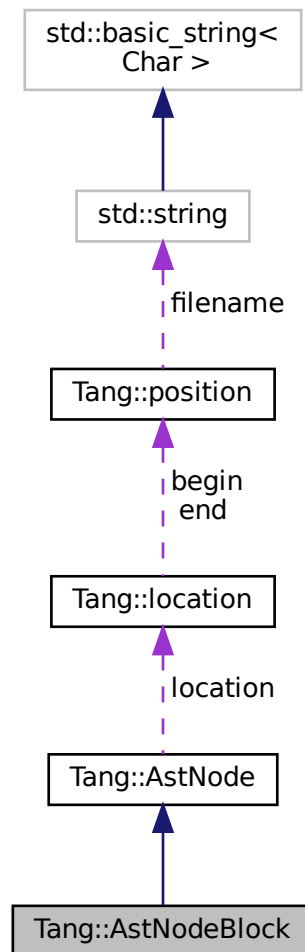
An [AstNode](#) that represents a code block.

```
#include <astNodeBlock.hpp>
```

Inheritance diagram for Tang::AstNodeBlock:



Collaboration diagram for Tang::AstNodeBlock:



Public Member Functions

- [AstNodeBlock](#) (const std::vector< std::shared_ptr< [AstNode](#) >> &statements, [Tang::location](#) location)
The constructor.
- virtual std::string [dump](#) (std::string indent="") const override
Return a string that describes the contents of the node.
- virtual void [compile](#) ([Tang::Program](#) &program) const override
Compile the ast of the provided [Tang::Program](#).
- virtual shared_ptr< [AstNode](#) > [makeCopy](#) () const override
Provide a copy of the [AstNode](#) (recursively, if appropriate).
- virtual void [compileIdentifiers](#) ([Program](#) &program) const override
Compile a list of all variables in the scope.

Protected Attributes

- [Tang::location](#) `location`

The location associated with this node.

5.4.1 Detailed Description

An [AstNode](#) that represents a code block.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 AstNodeBlock()

```
AstNodeBlock::AstNodeBlock (
    const std::vector< std::shared_ptr< AstNode >> & statements,
    Tang::location location )
```

The constructor.

Parameters

<i>statements</i>	The statements of the code block.
<i>location</i>	The location associated with the expression.

5.4.3 Member Function Documentation

5.4.3.1 compileIdentifiers()

```
void AstNodeBlock::compileIdentifiers (
    Program & program ) const [override], [virtual]
```

Compile a list of all variables in the scope.

Parameters

<i>program</i>	The Tang::Program that is being compiled.
----------------	---

Reimplemented from [Tang::AstNode](#).

5.4.3.2 makeCopy()

```
shared_ptr< AstNode > AstNodeBlock::makeCopy ( ) const [override], [virtual]
```

Provide a copy of the [AstNode](#) (recursively, if appropriate).

Returns

A pointer to a new [AstNode](#) that is a copy of the current [AstNode](#).

Reimplemented from [Tang::AstNode](#).

The documentation for this class was generated from the following files:

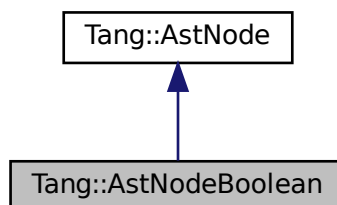
- include/[astNodeBlock.hpp](#)
- src/[astNodeBlock.cpp](#)

5.5 Tang::AstNodeBoolean Class Reference

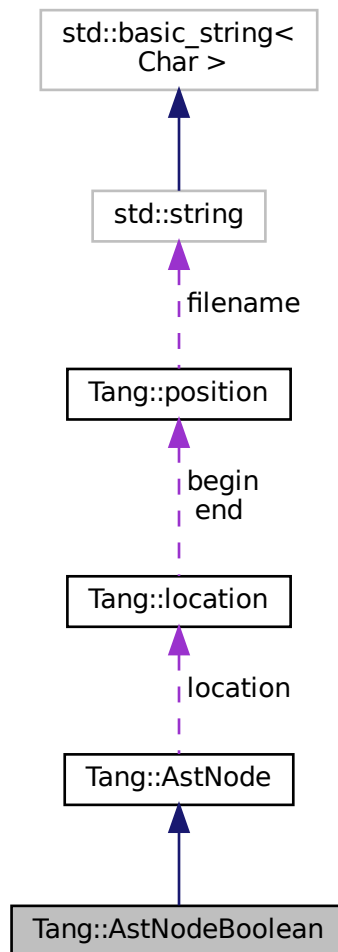
An [AstNode](#) that represents a boolean literal.

```
#include <astNodeBoolean.hpp>
```

Inheritance diagram for Tang::AstNodeBoolean:



Collaboration diagram for Tang::AstNodeBoolean:



Public Member Functions

- [AstNodeBoolean](#) (bool val, [Tang::location](#) location)
The constructor.
- virtual `std::string` [dump](#) (std::string indent="") const override
Return a string that describes the contents of the node.
- virtual void [compile](#) ([Tang::Program](#) &program) const override
Compile the ast of the provided [Tang::Program](#).
- virtual `shared_ptr< AstNode >` [makeCopy](#) () const override
Provide a copy of the [AstNode](#) (recursively, if appropriate).
- virtual void [compileIdentifiers](#) ([Program](#) &program) const
Compile a list of all variables in the scope.

Protected Attributes

- [Tang::location location](#)

The location associated with this node.

5.5.1 Detailed Description

An [AstNode](#) that represents a boolean literal.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 AstNodeBoolean()

```
AstNodeBoolean::AstNodeBoolean (
    bool val,
    Tang::location location )
```

The constructor.

Parameters

<i>val</i>	The boolean to represent.
<i>location</i>	The location associated with the expression.

5.5.3 Member Function Documentation

5.5.3.1 compileIdentifiers()

```
void AstNode::compileIdentifiers (
    Program & program ) const [virtual], [inherited]
```

Compile a list of all variables in the scope.

Parameters

<i>program</i>	The Tang::Program that is being compiled.
----------------	---

Reimplemented in [Tang::AstNodeIdentifier](#), [Tang::AstNodeBlock](#), [Tang::AstNodeBinary](#), and [Tang::AstNodeAssign](#).

5.5.3.2 makeCopy()

```
shared_ptr< AstNode > AstNodeBoolean::makeCopy ( ) const [override], [virtual]
```

Provide a copy of the [AstNode](#) (recursively, if appropriate).

Returns

A pointer to a new [AstNode](#) that is a copy of the current [AstNode](#).

Reimplemented from [Tang::AstNode](#).

The documentation for this class was generated from the following files:

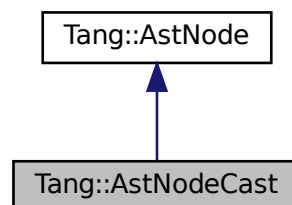
- include/[astNodeBoolean.hpp](#)
- src/[astNodeBoolean.cpp](#)

5.6 Tang::AstNodeCast Class Reference

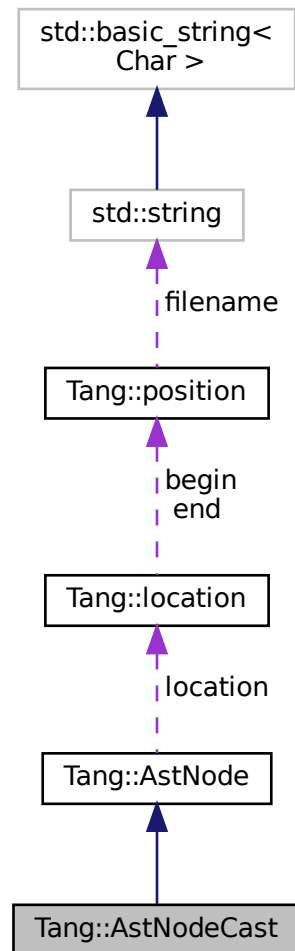
An [AstNode](#) that represents a typecast of an expression.

```
#include <astNodeCast.hpp>
```

Inheritance diagram for Tang::AstNodeCast:



Collaboration diagram for Tang::AstNodeCast:



Public Types

- enum `Type` { `Integer` , `Float` , `Boolean` }
- The possible types that can be cast to.*

Public Member Functions

- `AstNodeCast` (`Type` targetType, `shared_ptr< AstNode >` expression, `Tang::location` location)
The constructor.
- virtual `std::string` `dump` (`std::string` indent="") const override
Return a string that describes the contents of the node.
- virtual void `compile` (`Tang::Program` &program) const override
Compile the ast of the provided Tang::Program.
- virtual `shared_ptr< AstNode >` `makeCopy` () const override
Provide a copy of the AstNode (recursively, if appropriate).
- virtual void `compileIdentifiers` (`Program` &program) const
Compile a list of all variables in the scope.

Protected Attributes

- [Tang::location](#) `location`

The location associated with this node.

5.6.1 Detailed Description

An [AstNode](#) that represents a typecast of an expression.

5.6.2 Member Enumeration Documentation

5.6.2.1 Type

```
enum Tang::AstNodeCast::Type
```

The possible types that can be cast to.

Enumerator

Integer	Cast to a Tang::ComputedExpressionInteger .
Float	Cast to a Tang::ComputedExpressionFloat .
Boolean	Cast to a Tang::ComputedExpressionBoolean .

5.6.3 Constructor & Destructor Documentation

5.6.3.1 AstNodeCast()

```
AstNodeCast::AstNodeCast (
    Type targetType,
    shared_ptr< AstNode > expression,
    Tang::location location )
```

The constructor.

Parameters

<i>targetType</i>	The target type that the expression will be cast to.
<i>expression</i>	The expression to be typecast.
<i>location</i>	The location associated with this node.

5.6.4 Member Function Documentation

5.6.4.1 compileIdentifiers()

```
void AstNode::compileIdentifiers (
    Program & program ) const [virtual], [inherited]
```

Compile a list of all variables in the scope.

Parameters

<i>program</i>	The Tang::Program that is being compiled.
----------------	---

Reimplemented in [Tang::AstNodeIdentifier](#), [Tang::AstNodeBlock](#), [Tang::AstNodeBinary](#), and [Tang::AstNodeAssign](#).

5.6.4.2 makeCopy()

```
shared_ptr< AstNode > AstNodeCast::makeCopy ( ) const [override], [virtual]
```

Provide a copy of the [AstNode](#) (recursively, if appropriate).

Returns

A pointer to a new [AstNode](#) that is a copy of the current [AstNode](#).

Reimplemented from [Tang::AstNode](#).

The documentation for this class was generated from the following files:

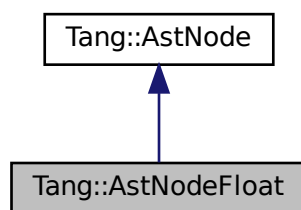
- [include/astNodeCast.hpp](#)
- [src/astNodeCast.cpp](#)

5.7 Tang::AstNodeFloat Class Reference

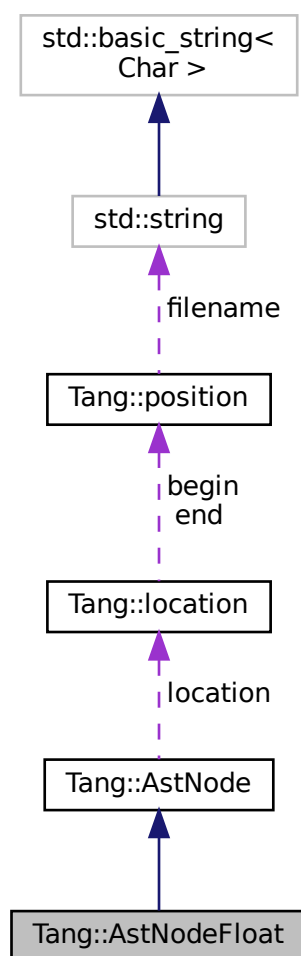
An [AstNode](#) that represents an float literal.

```
#include <astNodeFloat.hpp>
```

Inheritance diagram for Tang::AstNodeFloat:



Collaboration diagram for Tang::AstNodeFloat:



Public Member Functions

- [AstNodeFloat](#) (double number, [Tang::location](#) location)
The constructor.
- virtual std::string [dump](#) (std::string indent="") const override
Return a string that describes the contents of the node.
- virtual void [compile](#) ([Tang::Program](#) &program) const override
Compile the ast of the provided [Tang::Program](#).
- virtual shared_ptr< [AstNode](#) > [makeCopy](#) () const override
Provide a copy of the [AstNode](#) (recursively, if appropriate).
- virtual void [compileIdentifiers](#) ([Program](#) &program) const
Compile a list of all variables in the scope.

Protected Attributes

- [Tang::location](#) location
The location associated with this node.

5.7.1 Detailed Description

An [AstNode](#) that represents an float literal.

Integers are represented by the `long double` type, and so are limited in range by that of the underlying type.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 AstNodeFloat()

```
AstNodeFloat::AstNodeFloat (
    double number,
    Tang::location location )
```

The constructor.

Parameters

<i>number</i>	The number to represent.
<i>location</i>	The location associated with the expression.

5.7.3 Member Function Documentation

5.7.3.1 compileIdentifiers()

```
void AstNode::compileIdentifiers (
    Program & program ) const [virtual], [inherited]
```

Compile a list of all variables in the scope.

Parameters

<i>program</i>	The Tang::Program that is being compiled.
----------------	---

Reimplemented in [Tang::AstNodeIdentifier](#), [Tang::AstNodeBlock](#), [Tang::AstNodeBinary](#), and [Tang::AstNodeAssign](#).

5.7.3.2 makeCopy()

```
shared_ptr< AstNode > AstNodeFloat::makeCopy ( ) const [override], [virtual]
```

Provide a copy of the [AstNode](#) (recursively, if appropriate).

Returns

A pointer to a new [AstNode](#) that is a copy of the current [AstNode](#).

Reimplemented from [Tang::AstNode](#).

The documentation for this class was generated from the following files:

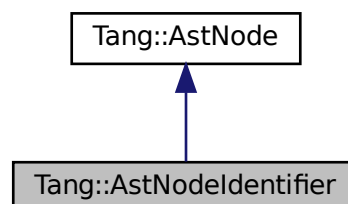
- include/[astNodeFloat.hpp](#)
- src/[astNodeFloat.cpp](#)

5.8 Tang::AstNodeIdentifier Class Reference

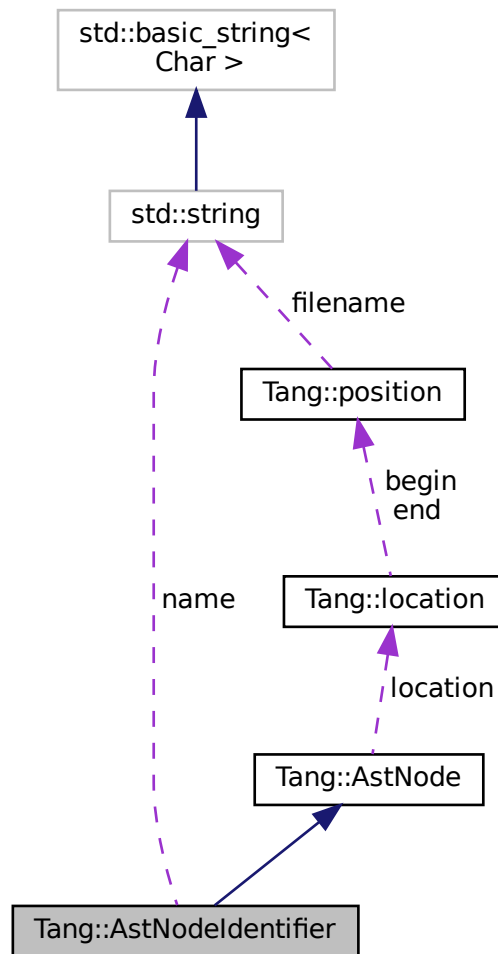
An [AstNode](#) that represents an identifier.

```
#include <astNodeIdentifier.hpp>
```

Inheritance diagram for Tang::AstNodeIdentifier:



Collaboration diagram for Tang::AstNodeIdentifier:



Public Member Functions

- [AstNodeIdentifier](#) (const std::string &name, Tang::location location)
The constructor.
- virtual std::string [dump](#) (std::string indent="") const override
Return a string that describes the contents of the node.
- virtual void [compile](#) (Tang::Program &program) const override
Compile the ast of the provided Tang::Program.
- virtual shared_ptr< [AstNode](#) > [makeCopy](#) () const override
Provide a copy of the AstNode (recursively, if appropriate).
- virtual void [compileIdentifiers](#) (Program &program) const override
Compile a list of all variables in the scope.

Public Attributes

- `std::string name`
The name of the identifier.

Protected Attributes

- `Tang::location location`
The location associated with this node.

5.8.1 Detailed Description

An `AstNode` that represents an identifier.

Identifier names are represented by a string.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 AstNodeIdentifier()

```
AstNodeIdentifier::AstNodeIdentifier (
    const std::string & name,
    Tang::location location )
```

The constructor.

Parameters

<i>name</i>	The name of the identifier
<i>location</i>	The location associated with the expression.

5.8.3 Member Function Documentation

5.8.3.1 compileIdentifiers()

```
void AstNodeIdentifier::compileIdentifiers (
    Program & program ) const [override], [virtual]
```

Compile a list of all variables in the scope.

Parameters

<i>program</i>	The Tang::Program that is being compiled.
----------------	---

Reimplemented from [Tang::AstNode](#).

5.8.3.2 makeCopy()

```
shared_ptr< AstNode > AstNodeIdentifier::makeCopy ( ) const [override], [virtual]
```

Provide a copy of the [AstNode](#) (recursively, if appropriate).

Returns

A pointer to a new [AstNode](#) that is a copy of the current [AstNode](#).

Reimplemented from [Tang::AstNode](#).

The documentation for this class was generated from the following files:

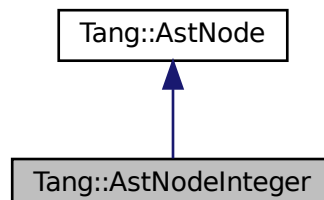
- include/[astNodeIdentifier.hpp](#)
- src/[astNodeIdentifier.cpp](#)

5.9 Tang::AstNodeInteger Class Reference

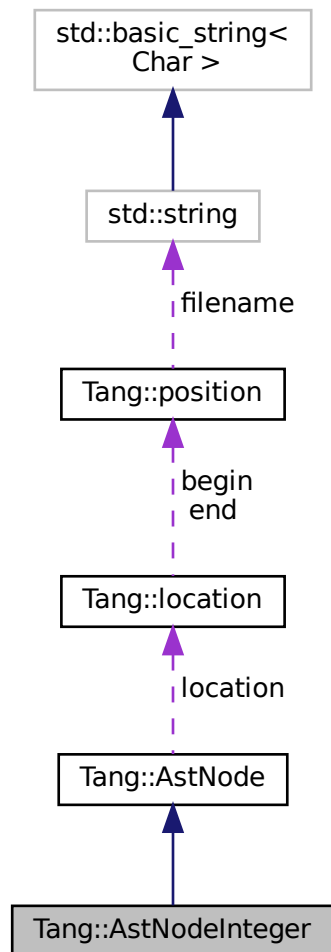
An [AstNode](#) that represents an integer literal.

```
#include <astNodeInteger.hpp>
```

Inheritance diagram for Tang::AstNodeInteger:



Collaboration diagram for Tang::AstNodeInteger:



Public Member Functions

- [AstNodeInteger](#) (int64_t number, [Tang::location](#) location)
The constructor.
- virtual std::string [dump](#) (std::string indent="") const override
Return a string that describes the contents of the node.
- virtual void [compile](#) ([Tang::Program](#) &program) const override
Compile the ast of the provided [Tang::Program](#).
- virtual shared_ptr< [AstNode](#) > [makeCopy](#) () const override
Provide a copy of the [AstNode](#) (recursively, if appropriate).
- virtual void [compileIdentifiers](#) ([Program](#) &program) const
Compile a list of all variables in the scope.

Protected Attributes

- [Tang::location](#) `location`

The location associated with this node.

5.9.1 Detailed Description

An [AstNode](#) that represents an integer literal.

Integers are represented by the `int64_t` type, and so are limited in range by that of the underlying type.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 AstNodeInteger()

```
AstNodeInteger::AstNodeInteger (
    int64_t number,
    Tang::location location )
```

The constructor.

Parameters

<i>number</i>	The number to represent.
<i>location</i>	The location associated with the expression.

5.9.3 Member Function Documentation

5.9.3.1 compileIdentifiers()

```
void AstNode::compileIdentifiers (
    Program & program ) const [virtual], [inherited]
```

Compile a list of all variables in the scope.

Parameters

<i>program</i>	The Tang::Program that is being compiled.
----------------	---

Reimplemented in [Tang::AstNodeIdentifier](#), [Tang::AstNodeBlock](#), [Tang::AstNodeBinary](#), and [Tang::AstNodeAssign](#).

5.9.3.2 makeCopy()

```
shared_ptr< AstNode > AstNodeInteger::makeCopy ( ) const [override], [virtual]
```

Provide a copy of the [AstNode](#) (recursively, if appropriate).

Returns

A pointer to a new [AstNode](#) that is a copy of the current [AstNode](#).

Reimplemented from [Tang::AstNode](#).

The documentation for this class was generated from the following files:

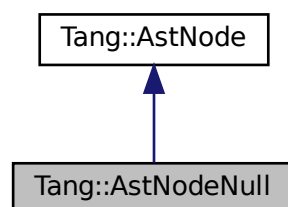
- include/[astNodeInteger.hpp](#)
- src/[astNodeInteger.cpp](#)

5.10 Tang::AstNodeNull Class Reference

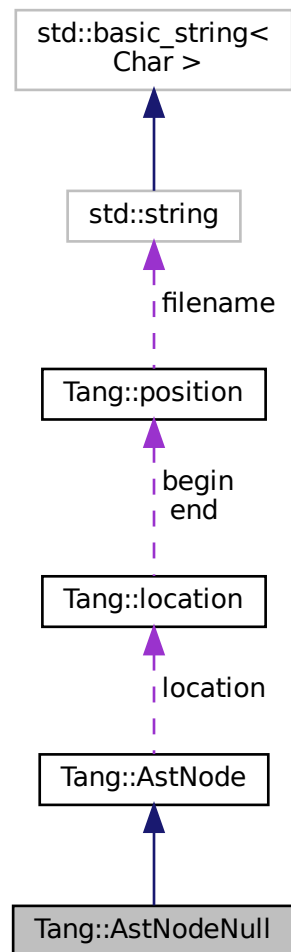
An [AstNode](#) that represents a NULL value.

```
#include <astNodeNull.hpp>
```

Inheritance diagram for Tang::AstNodeNull:



Collaboration diagram for Tang::AstNodeNull:



Public Member Functions

- [AstNodeNull](#) ([Tang::location](#) location)
The constructor.
- virtual std::string [dump](#) (std::string indent="") const override
Return a string that describes the contents of the node.
- virtual void [compile](#) ([Tang::Program](#) &program) const override
Compile the ast of the provided [Tang::Program](#).
- virtual shared_ptr< [AstNode](#) > [makeCopy](#) () const override
Provide a copy of the [AstNode](#) (recursively, if appropriate).
- virtual void [compileIdentifiers](#) ([Program](#) &program) const
Compile a list of all variables in the scope.

Protected Attributes

- [Tang::location](#) `location`

The location associated with this node.

5.10.1 Detailed Description

An [AstNode](#) that represents a NULL value.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 AstNodeNull()

```
AstNodeNull::AstNodeNull (
    Tang::location location )
```

The constructor.

Parameters

<i>location</i>	The location associated with the expression.
-----------------	--

5.10.3 Member Function Documentation

5.10.3.1 compileIdentifiers()

```
void AstNode::compileIdentifiers (
    Program & program ) const [virtual], [inherited]
```

Compile a list of all variables in the scope.

Parameters

<i>program</i>	The Tang::Program that is being compiled.
----------------	---

Reimplemented in [Tang::AstNodeIdentifier](#), [Tang::AstNodeBlock](#), [Tang::AstNodeBinary](#), and [Tang::AstNodeAssign](#).

5.10.3.2 makeCopy()

```
shared_ptr< AstNode > AstNodeNull::makeCopy ( ) const [override], [virtual]
```


Provide a copy of the [AstNode](#) (recursively, if appropriate).

Returns

A pointer to a new [AstNode](#) that is a copy of the current [AstNode](#).

Reimplemented from [Tang::AstNode](#).

The documentation for this class was generated from the following files:

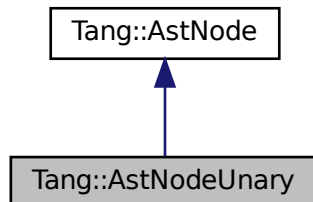
- include/[astNodeNull.hpp](#)
- src/[astNodeNull.cpp](#)

5.11 Tang::AstNodeUnary Class Reference

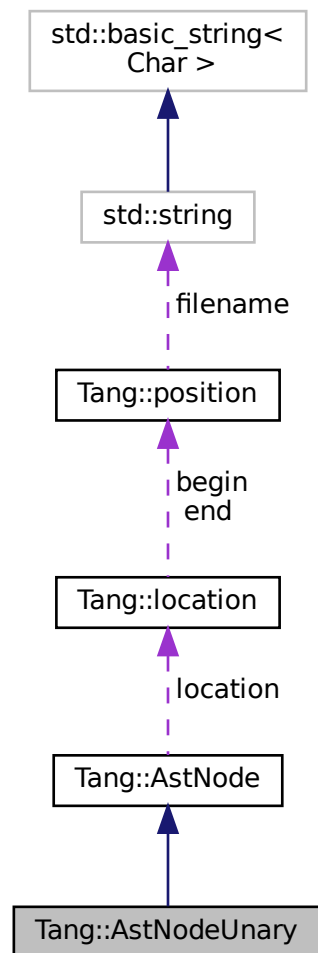
An [AstNode](#) that represents a unary negation.

```
#include <astNodeUnary.hpp>
```

Inheritance diagram for Tang::AstNodeUnary:



Collaboration diagram for Tang::AstNodeUnary:



Public Types

- enum `Operator` { `Negative` , `Not` }
- The type of operation.*

Public Member Functions

- `AstNodeUnary` (`Operator` op, `shared_ptr< AstNode >` operand, `Tang::location` location)
The constructor.
- virtual `std::string dump` (`std::string` indent="") const override
Return a string that describes the contents of the node.
- virtual void `compile` (`Tang::Program` &program) const override
Compile the ast of the provided Tang::Program.
- virtual `shared_ptr< AstNode >` `makeCopy` () const override
Provide a copy of the AstNode (recursively, if appropriate).
- virtual void `compileIdentifiers` (`Program` &program) const
Compile a list of all variables in the scope.

Protected Attributes

- [Tang::location location](#)

The location associated with this node.

5.11.1 Detailed Description

An [AstNode](#) that represents a unary negation.

5.11.2 Member Enumeration Documentation

5.11.2.1 Operator

```
enum Tang::AstNodeUnary::Operator
```

The type of operation.

Enumerator

Negative	Compute the negative (-).
Not	Compute the logical not (!).

5.11.3 Constructor & Destructor Documentation

5.11.3.1 AstNodeUnary()

```
AstNodeUnary::AstNodeUnary (
    Operator op,
    shared_ptr< AstNode > operand,
    Tang::location location )
```

The constructor.

Parameters

<i>op</i>	The Tang::AstNodeUnary::Operator to apply to the operand.
<i>operand</i>	The expression to be operated on.
<i>location</i>	The location associated with the expression.

5.11.4 Member Function Documentation

5.11.4.1 compileIdentifiers()

```
void AstNode::compileIdentifiers (
    Program & program ) const [virtual], [inherited]
```

Compile a list of all variables in the scope.

Parameters

<i>program</i>	The Tang::Program that is being compiled.
----------------	---

Reimplemented in [Tang::AstNodeIdentifier](#), [Tang::AstNodeBlock](#), [Tang::AstNodeBinary](#), and [Tang::AstNodeAssign](#).

5.11.4.2 makeCopy()

```
shared_ptr< AstNode > AstNodeUnary::makeCopy ( ) const [override], [virtual]
```

Provide a copy of the [AstNode](#) (recursively, if appropriate).

Returns

A pointer to a new [AstNode](#) that is a copy of the current [AstNode](#).

Reimplemented from [Tang::AstNode](#).

The documentation for this class was generated from the following files:

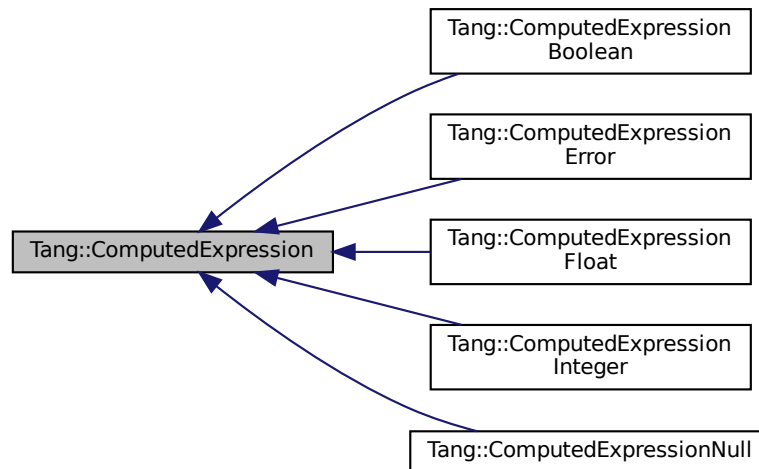
- [include/astNodeUnary.hpp](#)
- [src/astNodeUnary.cpp](#)

5.12 Tang::ComputedExpression Class Reference

Represents the result of a computation that has been executed.

```
#include <computedExpression.hpp>
```

Inheritance diagram for Tang::ComputedExpression:



Public Member Functions

- virtual `~ComputedExpression ()`
The object destructor.
- virtual `std::string dump () const`
Output the contents of the `ComputedExpression` as a string.
- virtual `ComputedExpression * makeCopy () const`
Make a copy of the `ComputedExpression` (recursively, if appropriate).
- virtual `bool is_equal (const int &val) const`
Check whether or not the computed expression is equal to another value.
- virtual `bool is_equal (const double &val) const`
Check whether or not the computed expression is equal to another value.
- virtual `bool is_equal (const bool &val) const`
Check whether or not the computed expression is equal to another value.
- virtual `bool is_equal (const Error &val) const`
Check whether or not the computed expression is equal to another value.
- virtual `bool is_equal (const std::nullptr_t &val) const`
Check whether or not the computed expression is equal to another value.
- virtual `GarbageCollected __add (const GarbageCollected &rhs) const`
Compute the result of adding this value and the supplied value.
- virtual `GarbageCollected __subtract (const GarbageCollected &rhs) const`
Compute the result of subtracting this value and the supplied value.
- virtual `GarbageCollected __multiply (const GarbageCollected &rhs) const`
Compute the result of multiplying this value and the supplied value.
- virtual `GarbageCollected __divide (const GarbageCollected &rhs) const`
Compute the result of dividing this value and the supplied value.
- virtual `GarbageCollected __modulo (const GarbageCollected &rhs) const`
Compute the result of moduloing this value and the supplied value.
- virtual `GarbageCollected __negative () const`

- Compute the result of negating this value.*
- virtual `GarbageCollected __not () const`
Compute the logical not of this value.
- virtual `GarbageCollected __lessThan (const GarbageCollected &rhs) const`
Compute the "less than" comparison.
- virtual `GarbageCollected __equal (const GarbageCollected &rhs) const`
Perform an equalit test.
- virtual `GarbageCollected __integer () const`
Perform a type cast to integer.
- virtual `GarbageCollected __float () const`
Perform a type cast to float.
- virtual `GarbageCollected __boolean () const`
Perform a type cast to boolean.

5.12.1 Detailed Description

Represents the result of a computation that has been executed.

5.12.2 Member Function Documentation

5.12.2.1 `__add()`

```
GarbageCollected ComputedExpression::__add (
    const GarbageCollected & rhs ) const [virtual]
```

Compute the result of adding this value and the supplied value.

Parameters

<i>rhs</i>	The <code>GarbageCollected</code> value to add to this.
------------	---

Returns

The result of the operation.

Reimplemented in `Tang::ComputedExpressionInteger`, `Tang::ComputedExpressionFloat`, and `Tang::ComputedExpressionError`.

5.12.2.2 `__boolean()`

```
GarbageCollected ComputedExpression::__boolean ( ) const [virtual]
```

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), [Tang::ComputedExpressionError](#), and [Tang::ComputedExpressionBoolean](#).

5.12.2.3 __divide()

```
GarbageCollected ComputedExpression::__divide (
    const GarbageCollected & rhs ) const [virtual]
```

Compute the result of dividing this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to divide this by.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.12.2.4 __equal()

```
GarbageCollected ComputedExpression::__equal (
    const GarbageCollected & rhs ) const [virtual]
```

Perform an equalit test.

Parameters

<i>rhs</i>	The GarbageCollected value to compare against.
------------	--

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionNull](#), [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), [Tang::ComputedExpressionError](#), and [Tang::ComputedExpressionBoolean](#).

5.12.2.5 `__float()`

```
GarbageCollected ComputedExpression::__float ( ) const [virtual]
```

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), [Tang::ComputedExpressionError](#), and [Tang::ComputedExpressionBoolean](#).

5.12.2.6 `__integer()`

```
GarbageCollected ComputedExpression::__integer ( ) const [virtual]
```

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), [Tang::ComputedExpressionError](#), and [Tang::ComputedExpressionBoolean](#).

5.12.2.7 `__lessThan()`

```
GarbageCollected ComputedExpression::__lessThan (
    const GarbageCollected & rhs ) const [virtual]
```

Compute the "less than" comparison.

Parameters

<i>rhs</i>	The GarbageCollected value to compare against.
------------	--

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.12.2.8 __modulo()

```
GarbageCollected ComputedExpression::__modulo (
    const GarbageCollected & rhs ) const [virtual]
```

Compute the result of moduloing this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to modulo this by.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionError](#).

5.12.2.9 __multiply()

```
GarbageCollected ComputedExpression::__multiply (
    const GarbageCollected & rhs ) const [virtual]
```

Compute the result of multiplying this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to multiply to this.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.12.2.10 __negative()

```
GarbageCollected ComputedExpression::__negative ( ) const [virtual]
```

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.12.2.11 __not()

```
GarbageCollected ComputedExpression::__not ( ) const [virtual]
```

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), [Tang::ComputedExpressionError](#), and [Tang::ComputedExpressionBoolean](#).

5.12.2.12 __subtract()

```
GarbageCollected ComputedExpression::__subtract (
    const GarbageCollected & rhs ) const [virtual]
```

Compute the result of subtracting this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to subtract from this.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.12.2.13 dump()

```
string ComputedExpression::dump ( ) const [virtual]
```

Output the contents of the [ComputedExpression](#) as a string.

Returns

A string representation of the computed expression.

Reimplemented in [Tang::ComputedExpressionNull](#), [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), [Tang::ComputedExpressionError](#), and [Tang::ComputedExpressionBoolean](#).

5.12.2.14 is_equal() [1/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const bool & val ) const [virtual]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionBoolean](#).

5.12.2.15 is_equal() [2/5]

```
virtual bool Tang::ComputedExpression::is_equal (  
    const double & val ) const [virtual]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.12.2.16 is_equal() [3/5]

```
virtual bool Tang::ComputedExpression::is_equal (  
    const Error & val ) const [virtual]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionError](#).

5.12.2.17 `is_equal()` [4/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const int & val ) const [virtual]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.12.2.18 `is_equal()` [5/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const std::nullptr_t & val ) const [virtual]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

5.12.2.19 `makeCopy()`

```
ComputedExpression * ComputedExpression::makeCopy ( ) const [virtual]
```

Make a copy of the [ComputedExpression](#) (recursively, if appropriate).

Returns

A pointer to the new [ComputedExpression](#).

Reimplemented in [Tang::ComputedExpressionNull](#), [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), [Tang::ComputedExpressionError](#), and [Tang::ComputedExpressionBoolean](#).

The documentation for this class was generated from the following files:

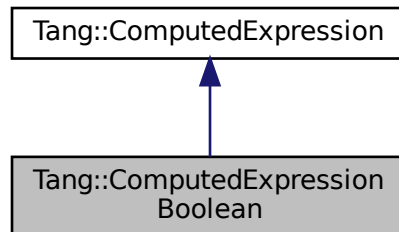
- [include/computedExpression.hpp](#)
- [src/computedExpression.cpp](#)

5.13 Tang::ComputedExpressionBoolean Class Reference

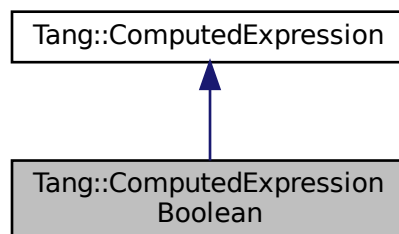
Represents an Boolean that is the result of a computation.

```
#include <computedExpressionBoolean.hpp>
```

Inheritance diagram for Tang::ComputedExpressionBoolean:



Collaboration diagram for Tang::ComputedExpressionBoolean:



Public Member Functions

- [ComputedExpressionBoolean](#) (bool val)
Construct an Boolean result.
- virtual std::string [dump](#) () const override
Output the contents of the [ComputedExpression](#) as a string.
- [ComputedExpression](#) * [makeCopy](#) () const override
Make a copy of the [ComputedExpression](#) (recursively, if appropriate).
- virtual bool [is_equal](#) (const bool &val) const override
Check whether or not the computed expression is equal to another value.
- virtual [GarbageCollected](#) [__not](#) () const override
Compute the logical not of this value.

- virtual `GarbageCollected __equal` (const `GarbageCollected` &rhs) const override
Perform an equalit test.
- virtual `GarbageCollected __integer` () const override
Perform a type cast to integer.
- virtual `GarbageCollected __float` () const override
Perform a type cast to float.
- virtual `GarbageCollected __boolean` () const override
Perform a type cast to boolean.
- virtual bool `is_equal` (const int &val) const
Check whether or not the computed expression is equal to another value.
- virtual bool `is_equal` (const double &val) const
Check whether or not the computed expression is equal to another value.
- virtual bool `is_equal` (const `Error` &val) const
Check whether or not the computed expression is equal to another value.
- virtual bool `is_equal` (const std::nullptr_t &val) const
Check whether or not the computed expression is equal to another value.
- virtual `GarbageCollected __add` (const `GarbageCollected` &rhs) const
Compute the result of adding this value and the supplied value.
- virtual `GarbageCollected __subtract` (const `GarbageCollected` &rhs) const
Compute the result of subtracting this value and the supplied value.
- virtual `GarbageCollected __multiply` (const `GarbageCollected` &rhs) const
Compute the result of multiplying this value and the supplied value.
- virtual `GarbageCollected __divide` (const `GarbageCollected` &rhs) const
Compute the result of dividing this value and the supplied value.
- virtual `GarbageCollected __modulo` (const `GarbageCollected` &rhs) const
Compute the result of moduloing this value and the supplied value.
- virtual `GarbageCollected __negative` () const
Compute the result of negating this value.
- virtual `GarbageCollected __lessThan` (const `GarbageCollected` &rhs) const
Compute the "less than" comparison.

5.13.1 Detailed Description

Represents an Boolean that is the result of a computation.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 ComputedExpressionBoolean()

```
ComputedExpressionBoolean::ComputedExpressionBoolean (
    bool val )
```

Construct an Boolean result.

Parameters

<i>val</i>	The boolean value.
------------	--------------------

5.13.3 Member Function Documentation

5.13.3.1 __add()

```
GarbageCollected ComputedExpression::__add (
    const GarbageCollected & rhs ) const [virtual], [inherited]
```

Compute the result of adding this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to add to this.
------------	--

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.13.3.2 __boolean()

```
GarbageCollected ComputedExpressionBoolean::__boolean ( ) const [override], [virtual]
```

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.13.3.3 __divide()

```
GarbageCollected ComputedExpression::__divide (
    const GarbageCollected & rhs ) const [virtual], [inherited]
```

Compute the result of dividing this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to divide this by.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.13.3.4 `__equal()`

```
GarbageCollected ComputedExpressionBoolean::__equal (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Perform an equalit test.

Parameters

<i>rhs</i>	The GarbageCollected value to compare against.
------------	--

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.13.3.5 `__float()`

```
GarbageCollected ComputedExpressionBoolean::__float ( ) const [override], [virtual]
```

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.13.3.6 __integer()

```
GarbageCollected ComputedExpressionBoolean::__integer ( ) const [override], [virtual]
```

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.13.3.7 __lessThan()

```
GarbageCollected ComputedExpression::__lessThan (
    const GarbageCollected & rhs ) const [virtual], [inherited]
```

Compute the "less than" comparison.

Parameters

<i>rhs</i>	The GarbageCollected value to compare against.
------------	--

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.13.3.8 __modulo()

```
GarbageCollected ComputedExpression::__modulo (
    const GarbageCollected & rhs ) const [virtual], [inherited]
```

Compute the result of moduloing this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to modulo this by.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionError](#).

5.13.3.9 __multiply()

```
GarbageCollected ComputedExpression::__multiply (
    const GarbageCollected & rhs ) const [virtual], [inherited]
```

Compute the result of multiplying this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to multiply to this.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.13.3.10 __negative()

```
GarbageCollected ComputedExpression::__negative ( ) const [virtual], [inherited]
```

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.13.3.11 __not()

```
GarbageCollected ComputedExpressionBoolean::__not ( ) const [override], [virtual]
```

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.13.3.12 __subtract()

```
GarbageCollected ComputedExpression::__subtract (
    const GarbageCollected & rhs ) const [virtual], [inherited]
```

Compute the result of subtracting this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to subtract from this.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.13.3.13 dump()

```
string ComputedExpressionBoolean::dump ( ) const [override], [virtual]
```

Output the contents of the [ComputedExpression](#) as a string.

Returns

A string representation of the computed expression.

Reimplemented from [Tang::ComputedExpression](#).

5.13.3.14 is_equal() [1/5]

```
bool ComputedExpressionBoolean::is_equal (
    const bool & val ) const [override], [virtual]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented from [Tang::ComputedExpression](#).

5.13.3.15 is_equal() [2/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const double & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.13.3.16 is_equal() [3/5]

```
virtual bool Tang::ComputedExpression::is_equal (  
    const Error & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionError](#).

5.13.3.17 is_equal() [4/5]

```
virtual bool Tang::ComputedExpression::is_equal (  
    const int & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.13.3.18 is_equal() [5/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const std::nullptr_t & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

5.13.3.19 makeCopy()

```
ComputedExpression * ComputedExpressionBoolean::makeCopy ( ) const [override], [virtual]
```

Make a copy of the [ComputedExpression](#) (recursively, if appropriate).

Returns

A pointer to the new [ComputedExpression](#).

Reimplemented from [Tang::ComputedExpression](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

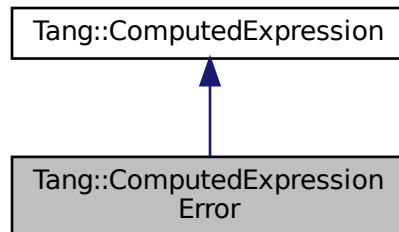
- [include/computedExpressionBoolean.hpp](#)
- [src/computedExpressionBoolean.cpp](#)

5.14 Tang::ComputedExpressionError Class Reference

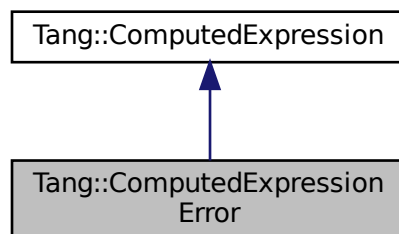
Represents a Runtime [Error](#).

```
#include <computedExpressionError.hpp>
```

Inheritance diagram for Tang::ComputedExpressionError:



Collaboration diagram for Tang::ComputedExpressionError:



Public Member Functions

- [ComputedExpressionError](#) ([Tang::Error](#) error)
Construct a Runtime [Error](#).
- virtual std::string [dump](#) () const override
Output the contents of the [ComputedExpression](#) as a string.
- [ComputedExpression](#) * [makeCopy](#) () const override
Make a copy of the [ComputedExpression](#) (recursively, if appropriate).
- virtual bool [is_equal](#) (const [Error](#) &val) const override
Check whether or not the computed expression is equal to another value.
- virtual [GarbageCollected](#) [__add](#) (const [GarbageCollected](#) &rhs) const override
Compute the result of adding this value and the supplied value.

- virtual `GarbageCollected __subtract` (const `GarbageCollected` &rhs) const override
Compute the result of subtracting this value and the supplied value.
- virtual `GarbageCollected __multiply` (const `GarbageCollected` &rhs) const override
Compute the result of multiplying this value and the supplied value.
- virtual `GarbageCollected __divide` (const `GarbageCollected` &rhs) const override
Compute the result of dividing this value and the supplied value.
- virtual `GarbageCollected __modulo` (const `GarbageCollected` &rhs) const override
Compute the result of moduloing this value and the supplied value.
- virtual `GarbageCollected __negative` () const override
Compute the result of negating this value.
- virtual `GarbageCollected __not` () const override
Compute the logical not of this value.
- virtual `GarbageCollected __lessThan` (const `GarbageCollected` &rhs) const override
Compute the "less than" comparison.
- virtual `GarbageCollected __equal` (const `GarbageCollected` &rhs) const override
Perform an equalit test.
- virtual `GarbageCollected __integer` () const override
Perform a type cast to integer.
- virtual `GarbageCollected __float` () const override
Perform a type cast to float.
- virtual `GarbageCollected __boolean` () const override
Perform a type cast to boolean.
- virtual bool `is_equal` (const int &val) const
Check whether or not the computed expression is equal to another value.
- virtual bool `is_equal` (const double &val) const
Check whether or not the computed expression is equal to another value.
- virtual bool `is_equal` (const bool &val) const
Check whether or not the computed expression is equal to another value.
- virtual bool `is_equal` (const std::nullptr_t &val) const
Check whether or not the computed expression is equal to another value.

5.14.1 Detailed Description

Represents a Runtime [Error](#).

5.14.2 Constructor & Destructor Documentation

5.14.2.1 ComputedExpressionError()

```
ComputedExpressionError::ComputedExpressionError (
    Tang::Error error )
```

Construct a Runtime [Error](#).

Parameters

<i>error</i>	The Tang::Error object.
--------------	---

5.14.3 Member Function Documentation

5.14.3.1 `__add()`

```
GarbageCollected ComputedExpressionError::__add (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the result of adding this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to add to this.
------------	--

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.2 `__boolean()`

```
GarbageCollected ComputedExpressionError::__boolean ( ) const [override], [virtual]
```

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.3 `__divide()`

```
GarbageCollected ComputedExpressionError::__divide (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the result of dividing this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to divide this by.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.4 `__equal()`

```
GarbageCollected ComputedExpressionError::__equal (
    const GarbageCollected & rhs ) const  [override], [virtual]
```

Perform an equalit test.

Parameters

<i>rhs</i>	The GarbageCollected value to compare against.
------------	--

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.5 `__float()`

```
GarbageCollected ComputedExpressionError::__float ( ) const  [override], [virtual]
```

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.6 `__integer()`

`GarbageCollected` `ComputedExpressionError::__integer () const [override], [virtual]`

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.7 `__lessThan()`

`GarbageCollected` `ComputedExpressionError::__lessThan (`
`const GarbageCollected & rhs) const [override], [virtual]`

Compute the "less than" comparison.

Parameters

<i>rhs</i>	The GarbageCollected value to compare against.
------------	--

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.8 `__modulo()`

`GarbageCollected` `ComputedExpressionError::__modulo (`
`const GarbageCollected & rhs) const [override], [virtual]`

Compute the result of moduloing this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to modulo this by.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.9 __multiply()

```
GarbageCollected ComputedExpressionError::__multiply (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the result of multiplying this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to multiply to this.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.10 __negative()

```
GarbageCollected ComputedExpressionError::__negative ( ) const [override], [virtual]
```

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.11 __not()

```
GarbageCollected ComputedExpressionError::__not ( ) const [override], [virtual]
```

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.12 __subtract()

```
GarbageCollected ComputedExpressionError::__subtract (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the result of subtracting this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to subtract from this.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.13 dump()

```
std::string ComputedExpressionError::dump ( ) const [override], [virtual]
```

Output the contents of the [ComputedExpression](#) as a string.

Returns

A string representation of the computed expression.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.14 is_equal() [1/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const bool & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionBoolean](#).

5.14.3.15 is_equal() [2/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const double & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.14.3.16 is_equal() [3/5]

```
bool ComputedExpressionError::is_equal (
    const Error & val ) const  [override], [virtual]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented from [Tang::ComputedExpression](#).

5.14.3.17 is_equal() [4/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const int & val ) const  [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.14.3.18 is_equal() [5/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const std::nullptr_t & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

5.14.3.19 makeCopy()

```
ComputedExpression * ComputedExpressionError::makeCopy ( ) const [override], [virtual]
```

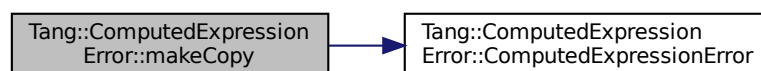
Make a copy of the [ComputedExpression](#) (recursively, if appropriate).

Returns

A pointer to the new [ComputedExpression](#).

Reimplemented from [Tang::ComputedExpression](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

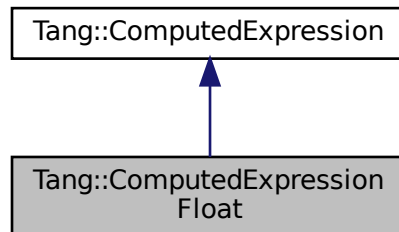
- [include/computedExpressionError.hpp](#)
- [src/computedExpressionError.cpp](#)

5.15 Tang::ComputedExpressionFloat Class Reference

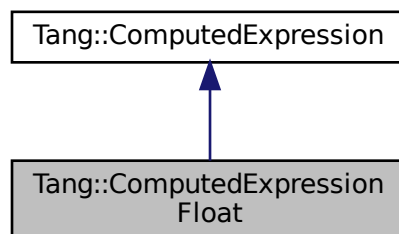
Represents a Float that is the result of a computation.

```
#include <computedExpressionFloat.hpp>
```

Inheritance diagram for Tang::ComputedExpressionFloat:



Collaboration diagram for Tang::ComputedExpressionFloat:



Public Member Functions

- [ComputedExpressionFloat](#) (double val)
Construct a Float result.
- virtual std::string [dump](#) () const override
Output the contents of the [ComputedExpression](#) as a string.
- [ComputedExpression](#) * [makeCopy](#) () const override
Make a copy of the [ComputedExpression](#) (recursively, if appropriate).
- virtual bool [is_equal](#) (const int &val) const override
Check whether or not the computed expression is equal to another value.
- virtual bool [is_equal](#) (const double &val) const override
Check whether or not the computed expression is equal to another value.

- virtual `GarbageCollected __add` (const `GarbageCollected &rhs`) const override
Compute the result of adding this value and the supplied value.
- virtual `GarbageCollected __subtract` (const `GarbageCollected &rhs`) const override
Compute the result of subtracting this value and the supplied value.
- virtual `GarbageCollected __multiply` (const `GarbageCollected &rhs`) const override
Compute the result of multiplying this value and the supplied value.
- virtual `GarbageCollected __divide` (const `GarbageCollected &rhs`) const override
Compute the result of dividing this value and the supplied value.
- virtual `GarbageCollected __negative` () const override
Compute the result of negating this value.
- virtual `GarbageCollected __not` () const override
Compute the logical not of this value.
- virtual `GarbageCollected __lessThan` (const `GarbageCollected &rhs`) const override
Compute the "less than" comparison.
- virtual `GarbageCollected __equal` (const `GarbageCollected &rhs`) const override
Perform an equality test.
- virtual `GarbageCollected __integer` () const override
Perform a type cast to integer.
- virtual `GarbageCollected __float` () const override
Perform a type cast to float.
- virtual `GarbageCollected __boolean` () const override
Perform a type cast to boolean.
- virtual bool `is_equal` (const bool &val) const
Check whether or not the computed expression is equal to another value.
- virtual bool `is_equal` (const `Error` &val) const
Check whether or not the computed expression is equal to another value.
- virtual bool `is_equal` (const std::nullptr_t &val) const
Check whether or not the computed expression is equal to another value.
- virtual `GarbageCollected __modulo` (const `GarbageCollected &rhs`) const
Compute the result of moduloing this value and the supplied value.

Friends

- class `ComputedExpressionInteger`

5.15.1 Detailed Description

Represents a Float that is the result of a computation.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 ComputedExpressionFloat()

```
ComputedExpressionFloat::ComputedExpressionFloat (
    double val )
```

Construct a Float result.

Parameters

<i>val</i>	The float value.
------------	------------------

5.15.3 Member Function Documentation

5.15.3.1 __add()

```
GarbageCollected ComputedExpressionFloat::__add (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the result of adding this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to add to this.
------------	--

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.2 __boolean()

```
GarbageCollected ComputedExpressionFloat::__boolean ( ) const [override], [virtual]
```

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.3 __divide()

```
GarbageCollected ComputedExpressionFloat::__divide (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the result of dividing this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to divide this by.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.4 `__equal()`

```
GarbageCollected ComputedExpressionFloat::__equal (
    const GarbageCollected & rhs ) const  [override], [virtual]
```

Perform an equalit test.

Parameters

<i>rhs</i>	The GarbageCollected value to compare against.
------------	--

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.5 `__float()`

```
GarbageCollected ComputedExpressionFloat::__float ( ) const  [override], [virtual]
```

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.6 __integer()

```
GarbageCollected ComputedExpressionFloat::__integer ( ) const [override], [virtual]
```

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.7 __lessThan()

```
GarbageCollected ComputedExpressionFloat::__lessThan (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the "less than" comparison.

Parameters

<i>rhs</i>	The GarbageCollected value to compare against.
------------	--

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.8 __modulo()

```
GarbageCollected ComputedExpression::__modulo (
    const GarbageCollected & rhs ) const [virtual], [inherited]
```

Compute the result of moduloing this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to modulo this by.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionError](#).

5.15.3.9 __multiply()

```
GarbageCollected ComputedExpressionFloat::__multiply (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the result of multiplying this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to multiply to this.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.10 __negative()

```
GarbageCollected ComputedExpressionFloat::__negative ( ) const [override], [virtual]
```

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.11 __not()

```
GarbageCollected ComputedExpressionFloat::__not ( ) const [override], [virtual]
```

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.12 __subtract()

```
GarbageCollected ComputedExpressionFloat::__subtract (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the result of subtracting this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to subtract from this.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.13 dump()

```
string ComputedExpressionFloat::dump ( ) const [override], [virtual]
```

Output the contents of the [ComputedExpression](#) as a string.

Returns

A string representation of the computed expression.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.14 is_equal() [1/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const bool & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionBoolean](#).

5.15.3.15 is_equal() [2/5]

```
bool ComputedExpressionFloat::is_equal (
    const double & val ) const [override], [virtual]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.16 is_equal() [3/5]

```
virtual bool Tang::ComputedExpression::is_equal (  
    const Error & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionError](#).

5.15.3.17 is_equal() [4/5]

```
bool ComputedExpressionFloat::is_equal (  
    const int & val ) const [override], [virtual]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented from [Tang::ComputedExpression](#).

5.15.3.18 is_equal() [5/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const std::nullptr_t & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

5.15.3.19 makeCopy()

```
ComputedExpression * ComputedExpressionFloat::makeCopy ( ) const [override], [virtual]
```

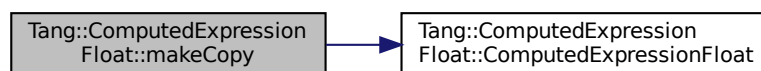
Make a copy of the [ComputedExpression](#) (recursively, if appropriate).

Returns

A pointer to the new [ComputedExpression](#).

Reimplemented from [Tang::ComputedExpression](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

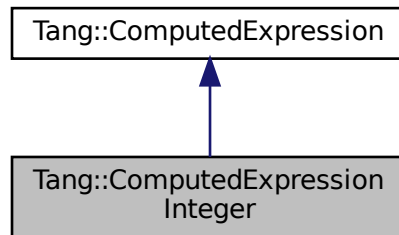
- [include/computedExpressionFloat.hpp](#)
- [src/computedExpressionFloat.cpp](#)

5.16 Tang::ComputedExpressionInteger Class Reference

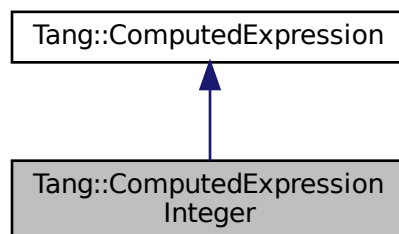
Represents an Integer that is the result of a computation.

```
#include <computedExpressionInteger.hpp>
```

Inheritance diagram for Tang::ComputedExpressionInteger:



Collaboration diagram for Tang::ComputedExpressionInteger:



Public Member Functions

- [ComputedExpressionInteger](#) (int64_t val)
Construct an Integer result.
- virtual std::string [dump](#) () const override
Output the contents of the [ComputedExpression](#) as a string.
- [ComputedExpression](#) * [makeCopy](#) () const override
Make a copy of the [ComputedExpression](#) (recursively, if appropriate).
- virtual bool [is_equal](#) (const int &val) const override
Check whether or not the computed expression is equal to another value.
- virtual bool [is_equal](#) (const double &val) const override
Check whether or not the computed expression is equal to another value.

- virtual [GarbageCollected](#) `__add` (const [GarbageCollected](#) &rhs) const override
Compute the result of adding this value and the supplied value.
- virtual [GarbageCollected](#) `__subtract` (const [GarbageCollected](#) &rhs) const override
Compute the result of subtracting this value and the supplied value.
- virtual [GarbageCollected](#) `__multiply` (const [GarbageCollected](#) &rhs) const override
Compute the result of multiplying this value and the supplied value.
- virtual [GarbageCollected](#) `__divide` (const [GarbageCollected](#) &rhs) const override
Compute the result of dividing this value and the supplied value.
- virtual [GarbageCollected](#) `__modulo` (const [GarbageCollected](#) &rhs) const override
Compute the result of moduloing this value and the supplied value.
- virtual [GarbageCollected](#) `__negative` () const override
Compute the result of negating this value.
- virtual [GarbageCollected](#) `__not` () const override
Compute the logical not of this value.
- virtual [GarbageCollected](#) `__lessThan` (const [GarbageCollected](#) &rhs) const override
Compute the "less than" comparison.
- virtual [GarbageCollected](#) `__equal` (const [GarbageCollected](#) &rhs) const override
Perform an equalit test.
- virtual [GarbageCollected](#) `__integer` () const override
Perform a type cast to integer.
- virtual [GarbageCollected](#) `__float` () const override
Perform a type cast to float.
- virtual [GarbageCollected](#) `__boolean` () const override
Perform a type cast to boolean.
- virtual bool `is_equal` (const bool &val) const
Check whether or not the computed expression is equal to another value.
- virtual bool `is_equal` (const [Error](#) &val) const
Check whether or not the computed expression is equal to another value.
- virtual bool `is_equal` (const std::nullptr_t &val) const
Check whether or not the computed expression is equal to another value.

Friends

- class [ComputedExpressionFloat](#)

5.16.1 Detailed Description

Represents an Integer that is the result of a computation.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 ComputedExpressionInteger()

```
ComputedExpressionInteger::ComputedExpressionInteger (
    int64_t val )
```

Construct an Integer result.

Parameters

<i>val</i>	The integer value.
------------	--------------------

5.16.3 Member Function Documentation

5.16.3.1 `__add()`

```
GarbageCollected ComputedExpressionInteger::__add (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the result of adding this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to add to this.
------------	--

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.2 `__boolean()`

```
GarbageCollected ComputedExpressionInteger::__boolean ( ) const [override], [virtual]
```

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.3 `__divide()`

```
GarbageCollected ComputedExpressionInteger::__divide (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the result of dividing this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to divide this by.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.4 `__equal()`

```
GarbageCollected ComputedExpressionInteger::__equal (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Perform an equalit test.

Parameters

<i>rhs</i>	The GarbageCollected value to compare against.
------------	--

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.5 `__float()`

```
GarbageCollected ComputedExpressionInteger::__float ( ) const [override], [virtual]
```

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.6 `__integer()`

```
GarbageCollected ComputedExpressionInteger::__integer ( ) const [override], [virtual]
```

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.7 `__lessThan()`

```
GarbageCollected ComputedExpressionInteger::__lessThan (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the "less than" comparison.

Parameters

<i>rhs</i>	The GarbageCollected value to compare against.
------------	--

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.8 `__modulo()`

```
GarbageCollected ComputedExpressionInteger::__modulo (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the result of moduloing this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to modulo this by.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.9 __multiply()

```
GarbageCollected ComputedExpressionInteger::__multiply (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the result of multiplying this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to multiply to this.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.10 __negative()

```
GarbageCollected ComputedExpressionInteger::__negative ( ) const [override], [virtual]
```

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.11 __not()

```
GarbageCollected ComputedExpressionInteger::__not ( ) const [override], [virtual]
```

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.12 __subtract()

```
GarbageCollected ComputedExpressionInteger::__subtract (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Compute the result of subtracting this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to subtract from this.
------------	---

Returns

The result of the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.13 dump()

```
string ComputedExpressionInteger::dump ( ) const [override], [virtual]
```

Output the contents of the [ComputedExpression](#) as a string.

Returns

A string representation of the computed expression.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.14 is_equal() [1/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const bool & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionBoolean](#).

5.16.3.15 is_equal() [2/5]

```
bool ComputedExpressionInteger::is_equal (
    const double & val ) const [override], [virtual]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.16 is_equal() [3/5]

```
virtual bool Tang::ComputedExpression::is_equal (  
    const Error & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionError](#).

5.16.3.17 is_equal() [4/5]

```
bool ComputedExpressionInteger::is_equal (  
    const int & val ) const [override], [virtual]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented from [Tang::ComputedExpression](#).

5.16.3.18 `is_equal()` [5/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const std::nullptr_t & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

5.16.3.19 `makeCopy()`

```
ComputedExpression * ComputedExpressionInteger::makeCopy ( ) const [override], [virtual]
```

Make a copy of the [ComputedExpression](#) (recursively, if appropriate).

Returns

A pointer to the new [ComputedExpression](#).

Reimplemented from [Tang::ComputedExpression](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

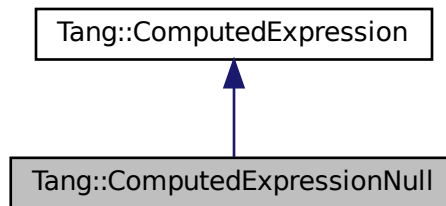
- [include/computedExpressionInteger.hpp](#)
- [src/computedExpressionInteger.cpp](#)

5.17 Tang::ComputedExpressionNull Class Reference

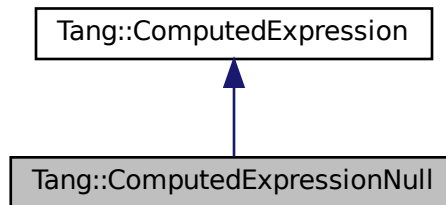
Represents an Null that is the result of a computation.

```
#include <computedExpressionNull.hpp>
```

Inheritance diagram for Tang::ComputedExpressionNull:



Collaboration diagram for Tang::ComputedExpressionNull:



Public Member Functions

- [ComputedExpressionNull](#) ()
Construct an Null result.
- virtual std::string [dump](#) () const override
Output the contents of the [ComputedExpression](#) as a string.
- [ComputedExpression](#) * [makeCopy](#) () const override
Make a copy of the [ComputedExpression](#) (recursively, if appropriate).
- bool [is_equal](#) (const nullptr_t &val) const override
- virtual [GarbageCollected](#) [__equal](#) (const [GarbageCollected](#) &rhs) const override
Perform an equalit test.
- virtual bool [is_equal](#) (const int &val) const
Check whether or not the computed expression is equal to another value.
- virtual bool [is_equal](#) (const double &val) const

- Check whether or not the computed expression is equal to another value.*
- virtual bool [is_equal](#) (const bool &val) const
- Check whether or not the computed expression is equal to another value.*
- virtual bool [is_equal](#) (const [Error](#) &val) const
- Check whether or not the computed expression is equal to another value.*
- virtual bool [is_equal](#) (const std::nullptr_t &val) const
- Check whether or not the computed expression is equal to another value.*
- virtual [GarbageCollected](#) [__add](#) (const [GarbageCollected](#) &rhs) const
- Compute the result of adding this value and the supplied value.*
- virtual [GarbageCollected](#) [__subtract](#) (const [GarbageCollected](#) &rhs) const
- Compute the result of subtracting this value and the supplied value.*
- virtual [GarbageCollected](#) [__multiply](#) (const [GarbageCollected](#) &rhs) const
- Compute the result of multiplying this value and the supplied value.*
- virtual [GarbageCollected](#) [__divide](#) (const [GarbageCollected](#) &rhs) const
- Compute the result of dividing this value and the supplied value.*
- virtual [GarbageCollected](#) [__modulo](#) (const [GarbageCollected](#) &rhs) const
- Compute the result of moduloing this value and the supplied value.*
- virtual [GarbageCollected](#) [__negative](#) () const
- Compute the result of negating this value.*
- virtual [GarbageCollected](#) [__not](#) () const
- Compute the logical not of this value.*
- virtual [GarbageCollected](#) [__lessThan](#) (const [GarbageCollected](#) &rhs) const
- Compute the "less than" comparison.*
- virtual [GarbageCollected](#) [__integer](#) () const
- Perform a type cast to integer.*
- virtual [GarbageCollected](#) [__float](#) () const
- Perform a type cast to float.*
- virtual [GarbageCollected](#) [__boolean](#) () const
- Perform a type cast to boolean.*

5.17.1 Detailed Description

Represents an Null that is the result of a computation.

5.17.2 Member Function Documentation

5.17.2.1 [__add\(\)](#)

```
GarbageCollected ComputedExpression::__add (
    const GarbageCollected & rhs ) const [virtual], [inherited]
```

Compute the result of adding this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to add to this.
------------	--

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.17.2.2 __boolean()

```
GarbageCollected ComputedExpression::__boolean ( ) const [virtual], [inherited]
```

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), [Tang::ComputedExpressionError](#), and [Tang::ComputedExpressionBoolean](#).

5.17.2.3 __divide()

```
GarbageCollected ComputedExpression::__divide (
    const GarbageCollected & rhs ) const [virtual], [inherited]
```

Compute the result of dividing this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to divide this by.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.17.2.4 __equal()

```
GarbageCollected ComputedExpressionNull::__equal (
    const GarbageCollected & rhs ) const [override], [virtual]
```

Perform an equalit test.

Parameters

<i>rhs</i>	The GarbageCollected value to compare against.
------------	--

Returns

The result of the the operation.

Reimplemented from [Tang::ComputedExpression](#).

5.17.2.5 __float()

```
GarbageCollected ComputedExpression::__float ( ) const [virtual], [inherited]
```

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), [Tang::ComputedExpressionError](#), and [Tang::ComputedExpressionBoolean](#).

5.17.2.6 __integer()

```
GarbageCollected ComputedExpression::__integer ( ) const [virtual], [inherited]
```

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), [Tang::ComputedExpressionError](#), and [Tang::ComputedExpressionBoolean](#).

5.17.2.7 __lessThan()

```
GarbageCollected ComputedExpression::__lessThan (
    const GarbageCollected & rhs ) const [virtual], [inherited]
```

Compute the "less than" comparison.

Parameters

<i>rhs</i>	The GarbageCollected value to compare against.
------------	--

Returns

The result of the the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.17.2.8 __modulo()

```
GarbageCollected ComputedExpression::__modulo (
    const GarbageCollected & rhs ) const [virtual], [inherited]
```

Compute the result of moduloing this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to modulo this by.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionError](#).

5.17.2.9 __multiply()

```
GarbageCollected ComputedExpression::__multiply (
    const GarbageCollected & rhs ) const [virtual], [inherited]
```

Compute the result of multiplying this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to multiply to this.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.17.2.10 __negative()

`GarbageCollected ComputedExpression::__negative () const [virtual], [inherited]`

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.17.2.11 __not()

`GarbageCollected ComputedExpression::__not () const [virtual], [inherited]`

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), [Tang::ComputedExpressionError](#), and [Tang::ComputedExpressionBoolean](#).

5.17.2.12 __subtract()

`GarbageCollected ComputedExpression::__subtract (
 const GarbageCollected & rhs) const [virtual], [inherited]`

Compute the result of subtracting this value and the supplied value.

Parameters

<i>rhs</i>	The GarbageCollected value to subtract from this.
------------	---

Returns

The result of the operation.

Reimplemented in [Tang::ComputedExpressionInteger](#), [Tang::ComputedExpressionFloat](#), and [Tang::ComputedExpressionError](#).

5.17.2.13 dump()

```
string ComputedExpressionNull::dump ( ) const [override], [virtual]
```

Output the contents of the [ComputedExpression](#) as a string.

Returns

A string representation of the computed expression.

Reimplemented from [Tang::ComputedExpression](#).

5.17.2.14 is_equal() [1/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const bool & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionBoolean](#).

5.17.2.15 is_equal() [2/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const double & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.17.2.16 `is_equal()` [3/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const Error & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionError](#).

5.17.2.17 `is_equal()` [4/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const int & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

Reimplemented in [Tang::ComputedExpressionInteger](#), and [Tang::ComputedExpressionFloat](#).

5.17.2.18 `is_equal()` [5/5]

```
virtual bool Tang::ComputedExpression::is_equal (
    const std::nullptr_t & val ) const [virtual], [inherited]
```

Check whether or not the computed expression is equal to another value.

Parameters

<i>val</i>	The value to compare against.
------------	-------------------------------

Returns

True if equal, false if not.

5.17.2.19 makeCopy()

```
ComputedExpression * ComputedExpressionNull::makeCopy ( ) const [override], [virtual]
```

Make a copy of the [ComputedExpression](#) (recursively, if appropriate).

Returns

A pointer to the new [ComputedExpression](#).

Reimplemented from [Tang::ComputedExpression](#).

The documentation for this class was generated from the following files:

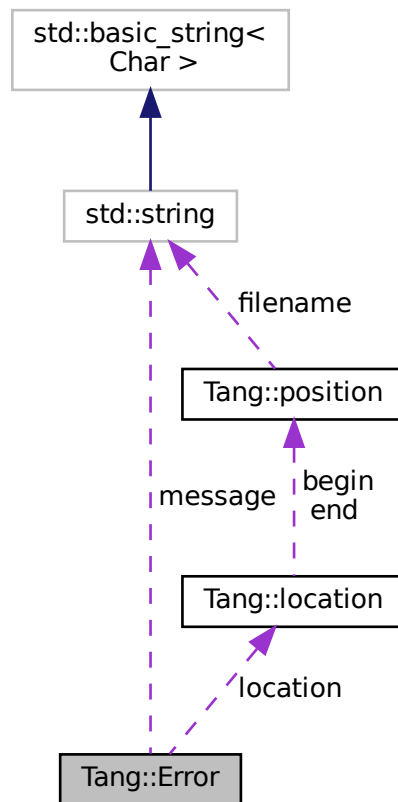
- include/[computedExpressionNull.hpp](#)
- src/[computedExpressionNull.cpp](#)

5.18 Tang::Error Class Reference

The [Error](#) class is used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error.

```
#include <error.hpp>
```

Collaboration diagram for Tang::Error:



Public Member Functions

- [Error](#) ()
Creates an empty error message.
- [Error](#) (std::string [message](#))
Creates an error message using the supplied error string and location.
- [Error](#) (std::string [message](#), [Tang::location](#) [location](#))
Creates an error message using the supplied error string and location.

Public Attributes

- std::string [message](#)
The error message as a string.
- [Tang::location](#) [location](#)
The location of the error.

Friends

- `std::ostream & operator<< (std::ostream &out, const Error &error)`
Add friendly output.

5.18.1 Detailed Description

The `Error` class is used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 Error() [1/2]

```
Tang::Error::Error (
    std::string message ) [inline]
```

Creates an error message using the supplied error string and location.

Parameters

<i>message</i>	The error message as a string.
----------------	--------------------------------

5.18.2.2 Error() [2/2]

```
Tang::Error::Error (
    std::string message,
    Tang::location location ) [inline]
```

Creates an error message using the supplied error string and location.

Parameters

<i>message</i>	The error message as a string.
<i>location</i>	The location of the error.

5.18.3 Friends And Related Function Documentation

5.18.3.1 operator<<

```
std::ostream& operator<< (
    std::ostream & out,
    const Error & error ) [friend]
```

Add friendly output.

Parameters

<i>out</i>	The output stream.
<i>error</i>	The Error object.

Returns

The output stream.

The documentation for this class was generated from the following files:

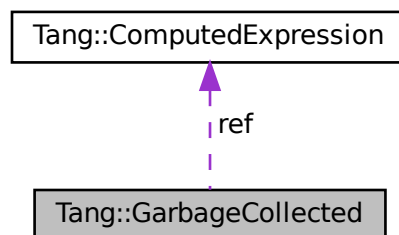
- [include/error.hpp](#)
- [src/error.cpp](#)

5.19 Tang::GarbageCollected Class Reference

A container that acts as a resource-counting garbage collector for the specified type.

```
#include <garbageCollected.hpp>
```

Collaboration diagram for Tang::GarbageCollected:



Public Member Functions

- [GarbageCollected](#) (const [GarbageCollected](#) &other)
Copy Constructor.
- [GarbageCollected](#) ([GarbageCollected](#) &&other)
Move Constructor.
- [GarbageCollected](#) & operator= (const [GarbageCollected](#) &other)
Copy Assignment.
- [GarbageCollected](#) & operator= ([GarbageCollected](#) &&other)
Move Assignment.
- [~GarbageCollected](#) ()
Destructor.
- [ComputedExpression](#) * operator-> () const
Access the tracked object as a pointer.
- [ComputedExpression](#) & operator* () const
Access the tracked object.
- bool operator== (const int &val) const
Compare the [GarbageCollected](#) tracked object with a supplied value.
- bool operator== (const double &val) const
Compare the [GarbageCollected](#) tracked object with a supplied value.
- bool operator== (const bool &val) const
Compare the [GarbageCollected](#) tracked object with a supplied value.
- bool operator== (const [Error](#) &val) const
Compare the [GarbageCollected](#) tracked object with a supplied value.
- bool operator== (const std::nullptr_t &null) const
Compare the [GarbageCollected](#) tracked object with a supplied value.
- [GarbageCollected](#) operator+ (const [GarbageCollected](#) &rhs) const
Perform an addition between two [GarbageCollected](#) values.
- [GarbageCollected](#) operator- (const [GarbageCollected](#) &rhs) const
Perform a subtraction between two [GarbageCollected](#) values.
- [GarbageCollected](#) operator* (const [GarbageCollected](#) &rhs) const
Perform a multiplication between two [GarbageCollected](#) values.
- [GarbageCollected](#) operator/ (const [GarbageCollected](#) &rhs) const
Perform a division between two [GarbageCollected](#) values.
- [GarbageCollected](#) operator% (const [GarbageCollected](#) &rhs) const
Perform a modulo between two [GarbageCollected](#) values.
- [GarbageCollected](#) operator- () const
Perform a negation on the [GarbageCollected](#) value.
- [GarbageCollected](#) operator! () const
Perform a logical not on the [GarbageCollected](#) value.
- [GarbageCollected](#) operator< (const [GarbageCollected](#) &rhs) const
Perform a < between two [GarbageCollected](#) values.
- [GarbageCollected](#) operator<= (const [GarbageCollected](#) &rhs) const
Perform a <= between two [GarbageCollected](#) values.
- [GarbageCollected](#) operator> (const [GarbageCollected](#) &rhs) const
Perform a > between two [GarbageCollected](#) values.
- [GarbageCollected](#) operator>= (const [GarbageCollected](#) &rhs) const
Perform a >= between two [GarbageCollected](#) values.
- [GarbageCollected](#) operator== (const [GarbageCollected](#) &rhs) const
Perform a == between two [GarbageCollected](#) values.
- [GarbageCollected](#) operator!= (const [GarbageCollected](#) &rhs) const
Perform a != between two [GarbageCollected](#) values.

Static Public Member Functions

- `template<class T, typename... Args>`
`static GarbageCollected make (Args... args)`
Creates a garbage-collected object of the specified type.

Protected Member Functions

- `GarbageCollected ()`
Constructs a garbage-collected object of the specified type.

Protected Attributes

- `size_t * count`
The count of references to the tracked object.
- `ComputedExpression * ref`
A reference to the tracked object.
- `std::function< void(void)> recycle`
A cleanup function to recycle the object.

Friends

- `std::ostream & operator<< (std::ostream &out, const GarbageCollected &gc)`
Add friendly output.

5.19.1 Detailed Description

A container that acts as a resource-counting garbage collector for the specified type.

Uses the [SingletonObjectPool](#) to created and recycle object memory. The container is not thread-safe.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 [GarbageCollected\(\)](#) [1/3]

```
Tang::GarbageCollected::GarbageCollected (
    const GarbageCollected & other ) [inline]
```

Copy Constructor.

Parameters

<i>The</i>	other GarbageCollected object to copy.
------------	--

5.19.2.2 GarbageCollected() [2/3]

```
Tang::GarbageCollected::GarbageCollected (
    GarbageCollected && other ) [inline]
```

Move Constructor.

Parameters

<i>The</i>	other GarbageCollected object to move.
------------	--

5.19.2.3 ~GarbageCollected()

```
Tang::GarbageCollected::~~GarbageCollected ( ) [inline]
```

Destructor.

Clean up the tracked object, if appropriate.

5.19.2.4 GarbageCollected() [3/3]

```
Tang::GarbageCollected::GarbageCollected ( ) [inline], [protected]
```

Constructs a garbage-collected object of the specified type.

It is private so that a [GarbageCollected](#) object can only be created using the [GarbageCollected::make\(\)](#) function.

Parameters

<i>variable</i>	The arguments to pass to the constructor of the specified type.
-----------------	---

5.19.3 Member Function Documentation

5.19.3.1 make()

```
template<class T , typename... Args>
static GarbageCollected Tang::GarbageCollected::make (
    Args... args ) [inline], [static]
```

Creates a garbage-collected object of the specified type.

Parameters

<i>variable</i>	The arguments to pass to the constructor of the specified type.
-----------------	---

Returns

A [GarbageCollected](#) object.

Here is the call graph for this function:

5.19.3.2 `operator"!")()`

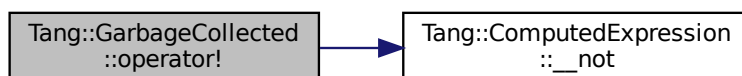
```
GarbageCollected GarbageCollected::operator! ( ) const
```

Perform a logical not on the [GarbageCollected](#) value.

Returns

The result of the operation.

Here is the call graph for this function:

5.19.3.3 `operator"!="(`

```
GarbageCollected GarbageCollected::operator!= (
    const GarbageCollected & rhs ) const
```

Perform a `!=` between two [GarbageCollected](#) values.

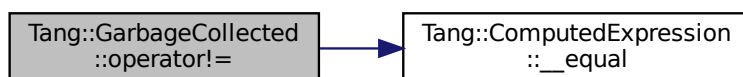
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:



5.19.3.4 operator%()

```
GarbageCollected GarbageCollected::operator% (
    const GarbageCollected & rhs ) const
```

Perform a modulo between two `GarbageCollected` values.

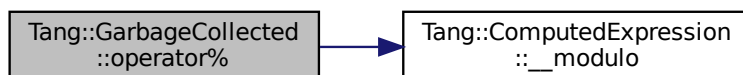
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:



5.19.3.5 operator*() [1/2]

```
ComputedExpression& Tang::GarbageCollected::operator* ( ) const [inline]
```

Access the tracked object.

Returns

A reference to the tracked object.

5.19.3.6 operator*() [2/2]

```
GarbageCollected GarbageCollected::operator* (
    const GarbageCollected & rhs ) const
```

Perform a multiplication between two [GarbageCollected](#) values.

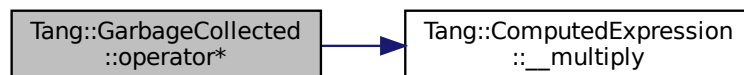
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:

**5.19.3.7 operator+()**

```
GarbageCollected GarbageCollected::operator+ (
    const GarbageCollected & rhs ) const
```

Perform an addition between two [GarbageCollected](#) values.

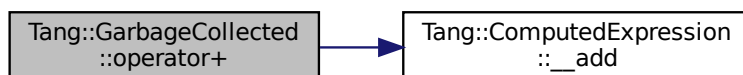
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:

5.19.3.8 `operator-()` [1/2]

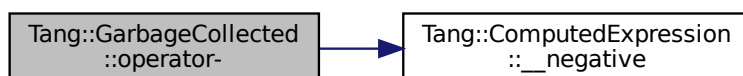
```
GarbageCollected GarbageCollected::operator- ( ) const
```

Perform a negation on the `GarbageCollected` value.

Returns

The result of the operation.

Here is the call graph for this function:

5.19.3.9 `operator-()` [2/2]

```
GarbageCollected GarbageCollected::operator- (
    const GarbageCollected & rhs ) const
```

Perform a subtraction between two `GarbageCollected` values.

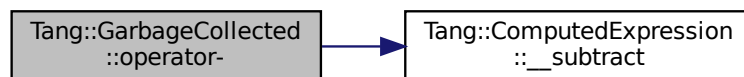
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:

**5.19.3.10 operator->()**

```
ComputedExpression* Tang::GarbageCollected::operator-> ( ) const [inline]
```

Access the tracked object as a pointer.

Returns

A pointer to the tracked object.

5.19.3.11 operator/()

```
GarbageCollected GarbageCollected::operator/ (
    const GarbageCollected & rhs ) const
```

Perform a division between two [GarbageCollected](#) values.

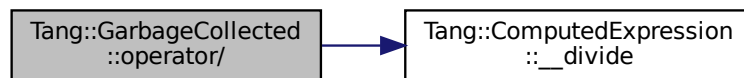
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:



5.19.3.12 `operator<()`

```
GarbageCollected Tang::GarbageCollected::operator< (
    const GarbageCollected & rhs ) const
```

Perform a `<` between two `GarbageCollected` values.

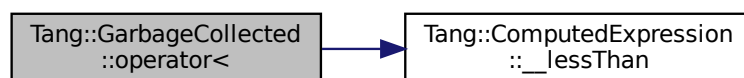
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:



5.19.3.13 `operator<=()`

```
GarbageCollected Tang::GarbageCollected::operator<= (
    const GarbageCollected & rhs ) const
```

Perform a `<=` between two `GarbageCollected` values.

Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

5.19.3.14 operator=() [1/2]

```
GarbageCollected& Tang::GarbageCollected::operator= (
    const GarbageCollected & other ) [inline]
```

Copy Assignment.

Parameters

<i>The</i>	other GarbageCollected object.
------------	--

Here is the call graph for this function:

**5.19.3.15 operator=()** [2/2]

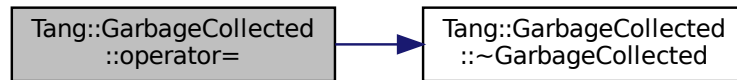
```
GarbageCollected& Tang::GarbageCollected::operator= (
    GarbageCollected && other ) [inline]
```

Move Assignment.

Parameters

<i>The</i>	other GarbageCollected object.
------------	--

Here is the call graph for this function:



5.19.3.16 operator==() [1/6]

```
bool GarbageCollected::operator== (
    const bool & val ) const
```

Compare the [GarbageCollected](#) tracked object with a supplied value.

Parameters

<i>val</i>	The value to compare the tracked object against.
------------	--

Returns

True if they are equal, false otherwise.

5.19.3.17 operator==() [2/6]

```
bool GarbageCollected::operator== (
    const double & val ) const
```

Compare the [GarbageCollected](#) tracked object with a supplied value.

Parameters

<i>val</i>	The value to compare the tracked object against.
------------	--

Returns

True if they are equal, false otherwise.

5.19.3.18 operator==() [3/6]

```
bool GarbageCollected::operator== (
    const Error & val ) const
```

Compare the [GarbageCollected](#) tracked object with a supplied value.

Parameters

<i>val</i>	The value to compare the tracked object against.
------------	--

Returns

True if they are equal, false otherwise.

5.19.3.19 operator==() [4/6]

```
GarbageCollected GarbageCollected::operator== (
    const GarbageCollected & rhs ) const
```

Perform a == between two [GarbageCollected](#) values.

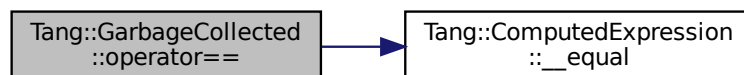
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:

**5.19.3.20 operator==() [5/6]**

```
bool GarbageCollected::operator== (
    const int & val ) const
```

Compare the [GarbageCollected](#) tracked object with a supplied value.

Parameters

<i>val</i>	The value to compare the tracked object against.
------------	--

Returns

True if they are equal, false otherwise.

5.19.3.21 operator==() [6/6]

```
bool GarbageCollected::operator==(
    const std::nullptr_t & null ) const
```

Compare the [GarbageCollected](#) tracked object with a supplied value.

Parameters

<i>val</i>	The value to compare the tracked object against.
------------	--

Returns

True if they are equal, false otherwise.

5.19.3.22 operator>()

```
GarbageCollected GarbageCollected::operator> (
    const GarbageCollected & rhs ) const
```

Perform a > between two [GarbageCollected](#) values.

Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

5.19.3.23 operator>=()

```
GarbageCollected GarbageCollected::operator>= (
    const GarbageCollected & rhs ) const
```

Perform a \geq between two [GarbageCollected](#) values.

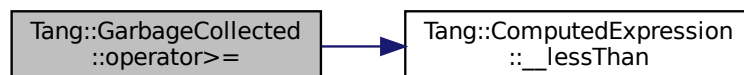
Parameters

<i>rhs</i>	The right hand side operand.
------------	------------------------------

Returns

The result of the operation.

Here is the call graph for this function:



5.19.4 Friends And Related Function Documentation

5.19.4.1 operator<<

```
std::ostream& operator<< (  
    std::ostream & out,  
    const GarbageCollected & gc ) [friend]
```

Add friendly output.

Parameters

<i>out</i>	The output stream.
<i>gc</i>	The GarbageCollected value.

Returns

The output stream.

The documentation for this class was generated from the following files:

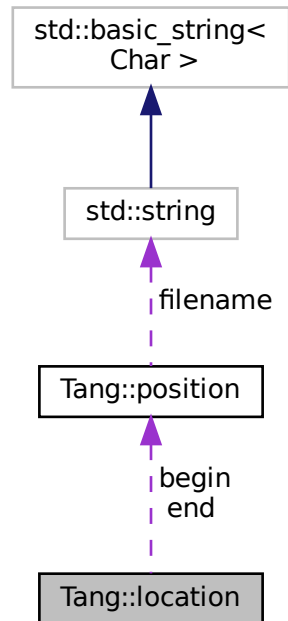
- [include/garbageCollected.hpp](#)
- [src/garbageCollected.cpp](#)

5.20 Tang::location Class Reference

Two points in a source file.

```
#include <location.hh>
```

Collaboration diagram for Tang::location:



Public Types

- typedef `position::filename_type filename_type`
Type for file name.
- typedef `position::counter_type counter_type`
Type for line and column numbers.

Public Member Functions

- `location` (const `position` &b, const `position` &e)
Construct a location from b to e.
- `location` (const `position` &p=`position`())
Construct a 0-width location in p.
- `location` (`filename_type` *f, `counter_type` l=1, `counter_type` c=1)
Construct a 0-width location in f, l, c.
- void `initialize` (`filename_type` *f=((void *) 0), `counter_type` l=1, `counter_type` c=1)
Initialization.

Line and Column related manipulators

- void `step` ()
Reset initial location to final location.
- void `columns` (`counter_type` count=1)
Extend the current location to the COUNT next columns.
- void `lines` (`counter_type` count=1)
Extend the current location to the COUNT next lines.

Public Attributes

- `position begin`
Beginning of the located region.
- `position end`
End of the located region.

5.20.1 Detailed Description

Two points in a source file.

The documentation for this class was generated from the following file:

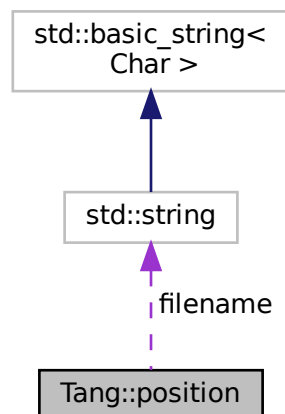
- build/generated/[location.hh](#)

5.21 Tang::position Class Reference

A point in a source file.

```
#include <location.hh>
```

Collaboration diagram for Tang::position:



Public Types

- typedef const std::string [filename_type](#)
Type for file name.
- typedef int [counter_type](#)
Type for line and column numbers.

Public Member Functions

- [position](#) ([filename_type](#) *f=((void *) 0), [counter_type](#) l=1, [counter_type](#) c=1)
Construct a position.
- void [initialize](#) ([filename_type](#) *fn=((void *) 0), [counter_type](#) l=1, [counter_type](#) c=1)
Initialization.

Line and Column related manipulators

- void [lines](#) ([counter_type](#) count=1)
(line related) Advance to the COUNT next lines.
- void [columns](#) ([counter_type](#) count=1)
(column related) Advance to the COUNT next columns.

Public Attributes

- [filename_type](#) * [filename](#)
File name to which this position refers.
- [counter_type](#) [line](#)
Current line number.
- [counter_type](#) [column](#)
Current column number.

5.21.1 Detailed Description

A point in a source file.

The documentation for this class was generated from the following file:

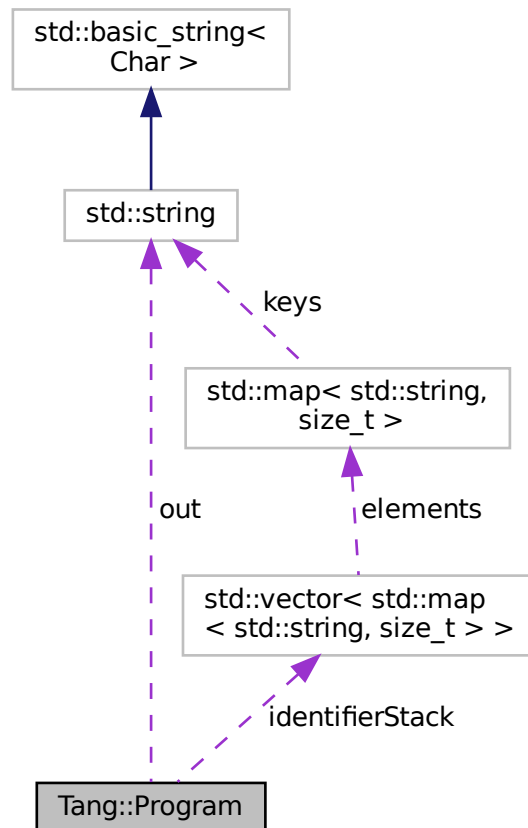
- build/generated/[location.hh](#)

5.22 Tang::Program Class Reference

Represents a compiled script or template that may be executed.

```
#include <program.hpp>
```

Collaboration diagram for Tang::Program:



Public Types

- enum `CodeType` { `Script` , `Template` }
Indicate the type of code that was supplied to the `Program`.

Public Member Functions

- `Program` (`std::string` code, `CodeType` codeType)
Create a compiled program using the provided code.
- `~Program` ()
The `Program` Destructor.

- [Program](#) (const [Program](#) &program)
The Copy Constructor.
- [Program](#) & [operator=](#) (const [Program](#) &program)
The Copy Assignment operator.
- [Program](#) ([Program](#) &&program)
The Move Constructor.
- [Program](#) & [operator=](#) ([Program](#) &&program)
The Move Assignment operator.
- std::string [getCode](#) () const
Get the code that was provided when the [Program](#) was created.
- std::optional< const std::shared_ptr< [AstNode](#) > > [getAst](#) () const
Get the AST that was generated by the parser.
- std::string [dumpBytecode](#) () const
Get the Opcodes of the compiled program, formatted like Assembly.
- std::optional< const [GarbageCollected](#) > [getResult](#) () const
Get the result of the [Program](#) execution, if it exists.
- void [addBytecode](#) (uint64_t)
Add a uint64_t to the Bytecode.
- [Program](#) & [execute](#) ()
Execute the program's Bytecode, and return the current [Program](#) object.

Public Attributes

- std::string [out](#)
The output of the program, resulting from the program execution.
- std::vector< std::map< std::string, size_t > > [identifierStack](#)
Stack of mappings of identifiers to their stack locations.

5.22.1 Detailed Description

Represents a compiled script or template that may be executed.

5.22.2 Member Enumeration Documentation

5.22.2.1 CodeType

```
enum Tang::Program::CodeType
```

Indicate the type of code that was supplied to the [Program](#).

Enumerator

Script	The code is pure Tang script, without any templating.
Template	The code is a template.

5.22.3 Constructor & Destructor Documentation

5.22.3.1 Program()

```
Program::Program (
    std::string code,
    Program::CodeType codeType )
```

Create a compiled program using the provided code.

Parameters

<i>code</i>	The code to be compiled.
<i>codeType</i>	Whether the code is a <i>Script</i> or <i>Template</i> .

5.22.4 Member Function Documentation

5.22.4.1 addBytecode()

```
void Program::addBytecode (
    uint64_t op )
```

Add a `uint64_t` to the Bytecode.

Parameters

<i>op</i>	The value to add to the Bytecode.
-----------	-----------------------------------

5.22.4.2 dumpBytecode()

```
string Program::dumpBytecode ( ) const
```

Get the Opcodes of the compiled program, formatted like Assembly.

Returns

A string containing the Opcode representation.

5.22.4.3 execute()

```
Program & Program::execute ( )
```

Execute the program's Bytecode, and return the current [Program](#) object.

Returns

The current [Program](#) object.

5.22.4.4 getAst()

```
optional< const shared_ptr< AstNode > > Program::getAst ( ) const
```

Get the AST that was generated by the parser.

The parser may have failed, so the return is an `optional<>` type. If the compilation failed, check `Program::error`.

Returns

A pointer to the AST, if it exists.

5.22.4.5 getCode()

```
string Program::getCode ( ) const
```

Get the code that was provided when the [Program](#) was created.

Returns

The source code from which the [Program](#) was created.

5.22.4.6 getResult()

```
optional< const GarbageCollected > Program::getResult ( ) const
```

Get the result of the [Program](#) execution, if it exists.

Returns

The result of the [Program](#) execution, if it exists.

The documentation for this class was generated from the following files:

- [include/program.hpp](#)
- [src/program-dumpBytecode.cpp](#)
- [src/program-execute.cpp](#)
- [src/program.cpp](#)

5.23 Tang::SingletonObjectPool< T > Class Template Reference

A thread-safe, singleton object pool of the designated type.

```
#include <singletonObjectPool.hpp>
```

Public Member Functions

- `T * get ()`
Request an uninitialized memory location from the pool for an object T.
- `void recycle (T *obj)`
Recycle a memory location for an object T.
- `~SingletonObjectPool ()`
Destructor.

Static Public Member Functions

- `static SingletonObjectPool< T > & getInstance ()`
Get the singleton instance of the object pool.

5.23.1 Detailed Description

```
template<class T>  
class Tang::SingletonObjectPool< T >
```

A thread-safe, singleton object pool of the designated type.

5.23.2 Member Function Documentation

5.23.2.1 `get()`

```
template<class T >  
T* Tang::SingletonObjectPool< T >::get ( ) [inline]
```

Request an uninitialized memory location from the pool for an object T.

Returns

An uninitialized memory location for an object T.

5.23.2.2 getInstance()

```
template<class T >
static SingletonObjectPool<T>& Tang::SingletonObjectPool< T >::getInstance ( ) [inline],
[static]
```

Get the singleton instance of the object pool.

Returns

The singleton instance of the object pool.

5.23.2.3 recycle()

```
template<class T >
void Tang::SingletonObjectPool< T >::recycle (
    T * obj ) [inline]
```

Recycle a memory location for an object T.

Parameters

<i>obj</i>	The memory location to recycle.
------------	---------------------------------

The documentation for this class was generated from the following file:

- [include/singletonObjectPool.hpp](#)

5.24 Tang::TangBase Class Reference

The base class for the Tang programming language.

```
#include <tangBase.hpp>
```

Public Member Functions

- [TangBase](#) ()
The constructor.
- [Program compileScript](#) (std::string script)
Compile the provided source code as a script and return a [Program](#).

5.24.1 Detailed Description

The base class for the Tang programming language.

This class is the fundamental starting point to compile and execute a Tang program. It may be considered in three parts:

1. It acts as an extendable interface through which additional "library" functions can be added to the language. It is intentionally designed that each instance of [TangBase](#) will have its own library functions.
2. It provides methods to compile scripts and templates, resulting in a [Program](#) object.
3. The [Program](#) object may then be executed, providing instance-specific context information (*i.e.*, state).

5.24.2 Constructor & Destructor Documentation

5.24.2.1 TangBase()

```
TangBase::TangBase ( )
```

The constructor.

Isn't it glorious.

5.24.3 Member Function Documentation

5.24.3.1 compileScript()

```
Program TangBase::compileScript (
    std::string script )
```

Compile the provided source code as a script and return a [Program](#).

Parameters

<i>script</i>	The Tang script to be compiled.
---------------	---------------------------------

Returns

The [Program](#) object representing the compiled script.

The documentation for this class was generated from the following files:

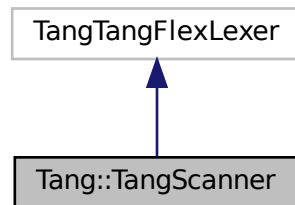
- include/[tangBase.hpp](#)
- src/[tangBase.cpp](#)

5.25 Tang::TangScanner Class Reference

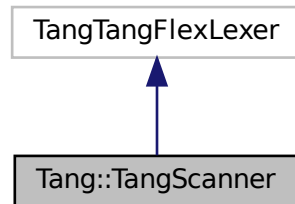
The Flex lexer class for the main Tang language.

```
#include <tangScanner.hpp>
```

Inheritance diagram for Tang::TangScanner:



Collaboration diagram for Tang::TangScanner:



Public Member Functions

- [TangScanner](#) (std::istream &arg_yyin, std::ostream &arg_yyout)
The constructor for the Scanner.
- virtual Tang::TangParser::symbol_type [get_next_token](#) ()
A pass-through function that we supply so that we can provide a Bison 3 token return type instead of the `int` that is returned by the default class configuration.

5.25.1 Detailed Description

The Flex lexer class for the main Tang language.

Flex requires that our lexer class inherit from yyFlexLexer, an "intermediate" class whose real name is "TangTangFlexLexer". We are subclassing it so that we can override the return type of [get_next_token\(\)](#), for compatibility with Bison 3 tokens.

5.25.2 Constructor & Destructor Documentation

5.25.2.1 TangScanner()

```
Tang::TangScanner::TangScanner (
    std::istream & arg_yyin,
    std::ostream & arg_yyout ) [inline]
```

The constructor for the Scanner.

The design of the Flex lexer is to tokenize the contents of an input stream, and to write any error messages to an output stream. In our implementation, however, errors are returned differently, so the output stream is never used. It's presence is retained, however, in case it is needed in the future.

For now, the general approach should be to supply the input as a string stream, and to use `std::cout` as the output.

Parameters

<i>arg_yyin</i>	The input stream to be tokenized
<i>arg_yyout</i>	The output stream (not currently used)

5.25.3 Member Function Documentation

5.25.3.1 get_next_token()

```
virtual Tang::TangParser::symbol_type Tang::TangScanner::get_next_token ( ) [virtual]
```

A pass-through function that we supply so that we can provide a Bison 3 token return type instead of the `int` that is returned by the default class configuration.

Returns

A Bison 3 token representing the lexeme that was recognized.

The documentation for this class was generated from the following file:

- include/[tangScanner.hpp](#)

Chapter 6

File Documentation

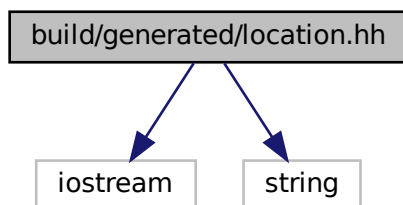
6.1 build/generated/location.hh File Reference

Define the Tang ::location class.

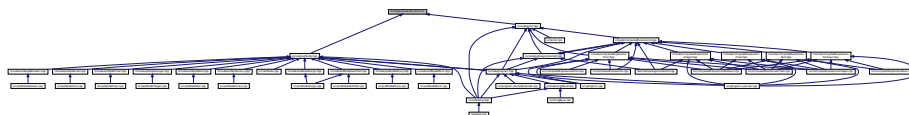
```
#include <iostream>
```

```
#include <string>
```

Include dependency graph for location.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::position](#)
A point in a source file.
- class [Tang::location](#)
Two points in a source file.

Macros

- `#define YY_NULLPTR ((void*)0)`

Functions

- position & [Tang::operator+=](#) (position &res, position::counter_type width)
Add width columns, in place.
- position [Tang::operator+](#) (position res, position::counter_type width)
Add width columns.
- position & [Tang::operator-=](#) (position &res, position::counter_type width)
Subtract width columns, in place.
- position [Tang::operator-](#) (position res, position::counter_type width)
Subtract width columns.
- template<typename YYChar >
std::basic_ostream< YYChar > & [Tang::operator<<](#) (std::basic_ostream< YYChar > &ostr, const position &pos)
Intercept output stream redirection.
- location & [Tang::operator+=](#) (location &res, const location &end)
Join two locations, in place.
- location [Tang::operator+](#) (location res, const location &end)
Join two locations.
- location & [Tang::operator+=](#) (location &res, location::counter_type width)
Add width columns to the end position, in place.
- location [Tang::operator+](#) (location res, location::counter_type width)
Add width columns to the end position.
- location & [Tang::operator-=](#) (location &res, location::counter_type width)
Subtract width columns to the end position, in place.
- location [Tang::operator-](#) (location res, location::counter_type width)
Subtract width columns to the end position.
- template<typename YYChar >
std::basic_ostream< YYChar > & [Tang::operator<<](#) (std::basic_ostream< YYChar > &ostr, const location &loc)
Intercept output stream redirection.

6.1.1 Detailed Description

Define the Tang ::location class.

6.1.2 Function Documentation

6.1.2.1 operator<<() [1/2]

```
template<typename YYChar >
std::basic_ostream<YYChar>& Tang::operator<< (
    std::basic_ostream< YYChar > & ostr,
    const location & loc )
```

Intercept output stream redirection.

Parameters

<i>ostr</i>	the destination output stream
<i>loc</i>	a reference to the location to redirect

Avoid duplicate information.

6.1.2.2 operator<<() [2/2]

```
template<typename YYChar >
std::basic_ostream<YYChar>& Tang::operator<< (
    std::basic_ostream< YYChar > & ostr,
    const position & pos )
```

Intercept output stream redirection.

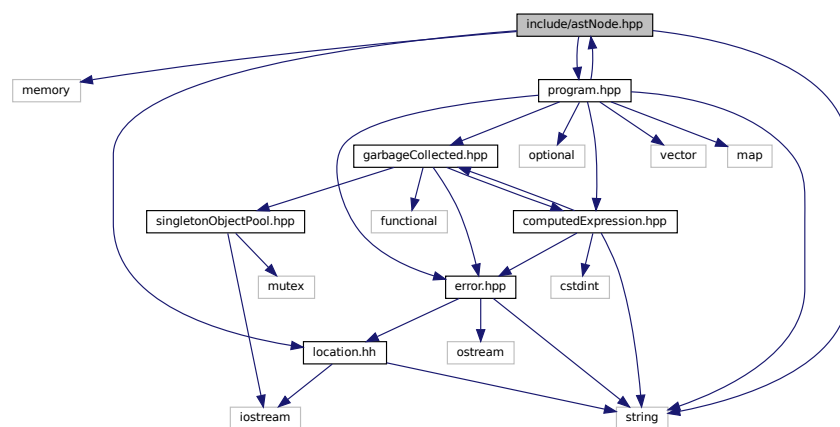
Parameters

<i>ostr</i>	the destination output stream
<i>pos</i>	a reference to the position to redirect

6.2 include/astNode.hpp File Reference

Declare the [Tang::AstNode](#) base class.

```
#include <memory>
#include <string>
#include "location.hh"
#include "program.hpp"
Include dependency graph for astNode.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::AstNode](#)
Base class for representing nodes of an Abstract Syntax Tree (AST).

6.2.1 Detailed Description

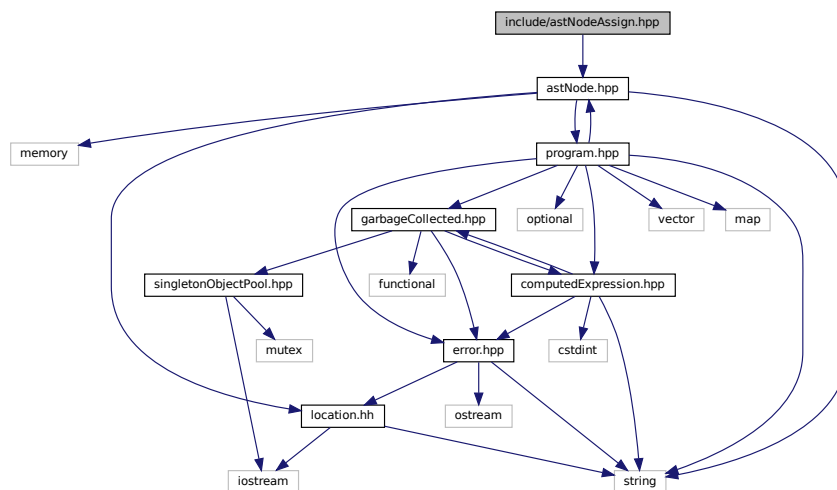
Declare the [Tang::AstNode](#) base class.

6.3 include/astNodeAssign.hpp File Reference

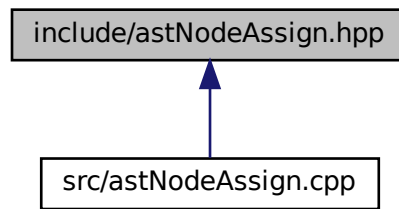
Declare the [Tang::AstNodeAssign](#) class.

```
#include "astNode.hpp"
```

Include dependency graph for astNodeAssign.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::AstNodeAssign](#)
An [AstNode](#) that represents a binary expression.

6.3.1 Detailed Description

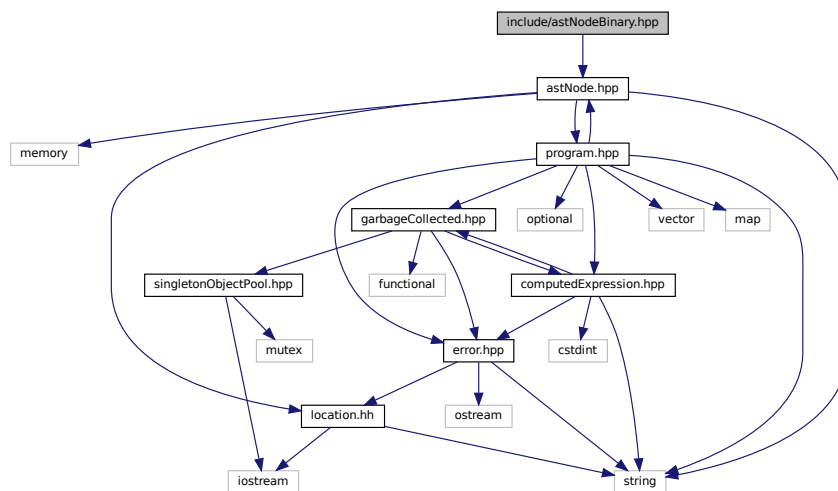
Declare the [Tang::AstNodeAssign](#) class.

6.4 include/astNodeBinary.hpp File Reference

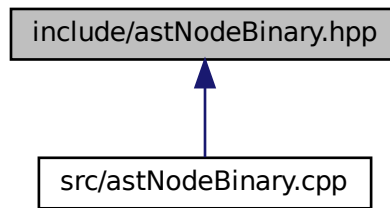
Declare the [Tang::AstNodeBinary](#) class.

```
#include "astNode.hpp"
```

Include dependency graph for astNodeBinary.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::AstNodeBinary](#)
An [AstNode](#) that represents a binary expression.

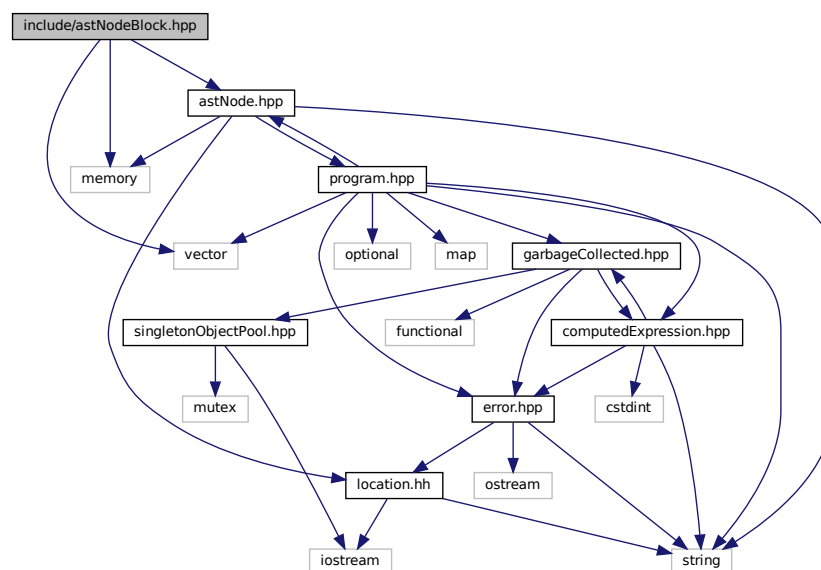
6.4.1 Detailed Description

Declare the [Tang::AstNodeBinary](#) class.

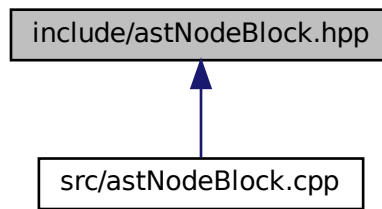
6.5 include/astNodeBlock.hpp File Reference

Declare the [Tang::AstNodeBlock](#) class.

```
#include <vector>
#include <memory>
#include "astNode.hpp"
Include dependency graph for astNodeBlock.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::AstNodeBlock](#)
An [AstNode](#) that represents a code block.

6.5.1 Detailed Description

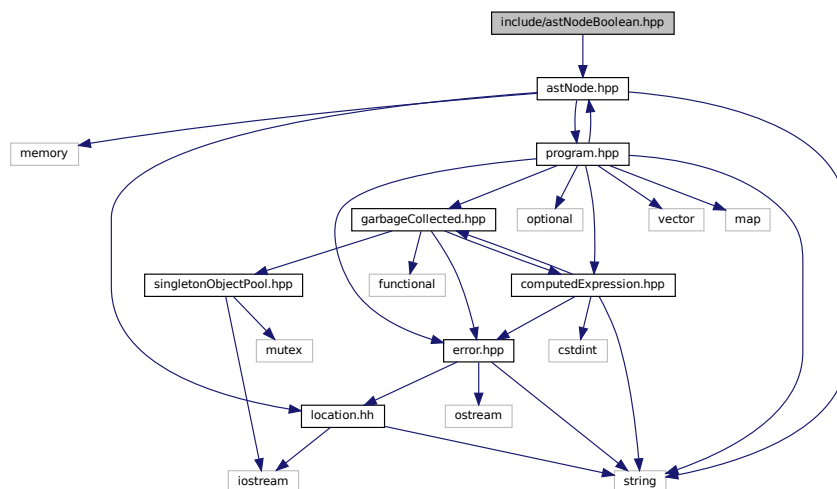
Declare the [Tang::AstNodeBlock](#) class.

6.6 include/astNodeBoolean.hpp File Reference

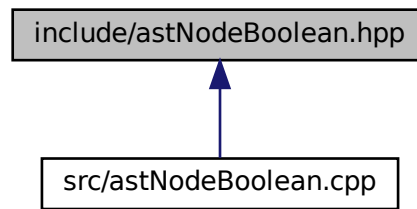
Declare the [Tang::AstNodeBoolean](#) class.

```
#include "astNode.hpp"
```

Include dependency graph for astNodeBoolean.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::AstNodeBoolean](#)
An [AstNode](#) that represents a boolean literal.

6.6.1 Detailed Description

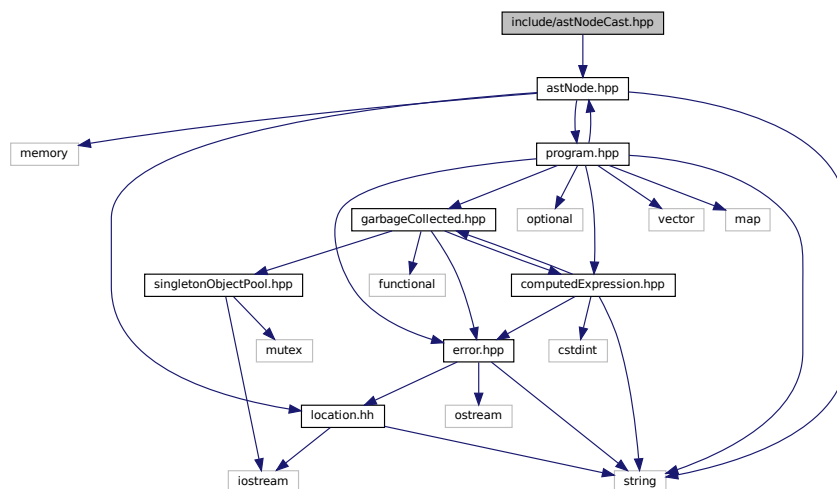
Declare the [Tang::AstNodeBoolean](#) class.

6.7 include/astNodeCast.hpp File Reference

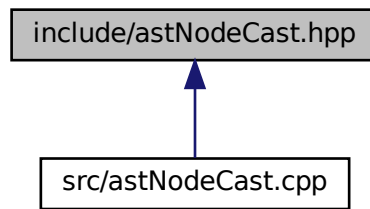
Declare the [Tang::AstNodeCast](#) class.

```
#include "astNode.hpp"
```

Include dependency graph for astNodeCast.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::AstNodeCast](#)
An [AstNode](#) that represents a typecast of an expression.

6.7.1 Detailed Description

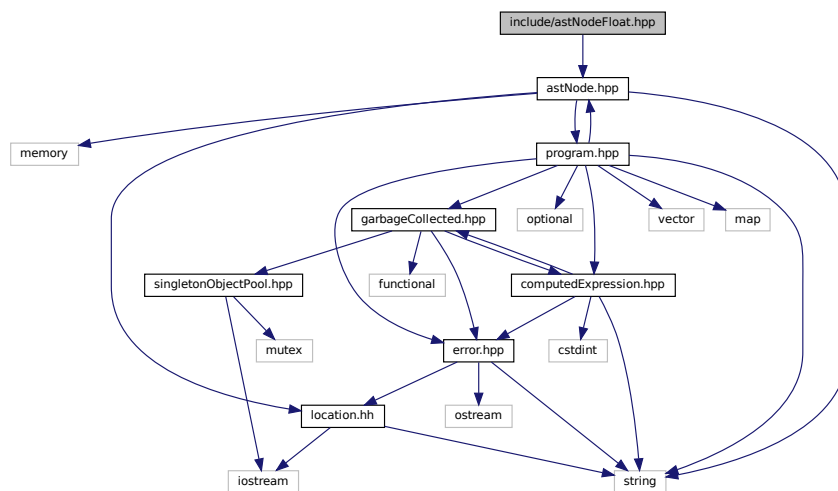
Declare the [Tang::AstNodeCast](#) class.

6.8 include/astNodeFloat.hpp File Reference

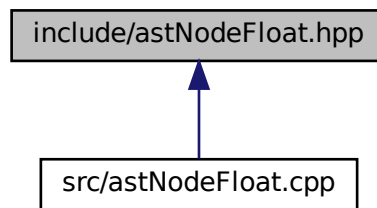
Declare the [Tang::AstNodeFloat](#) class.

```
#include "astNode.hpp"
```

Include dependency graph for astNodeFloat.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::AstNodeFloat](#)
An [AstNode](#) that represents an float literal.

6.8.1 Detailed Description

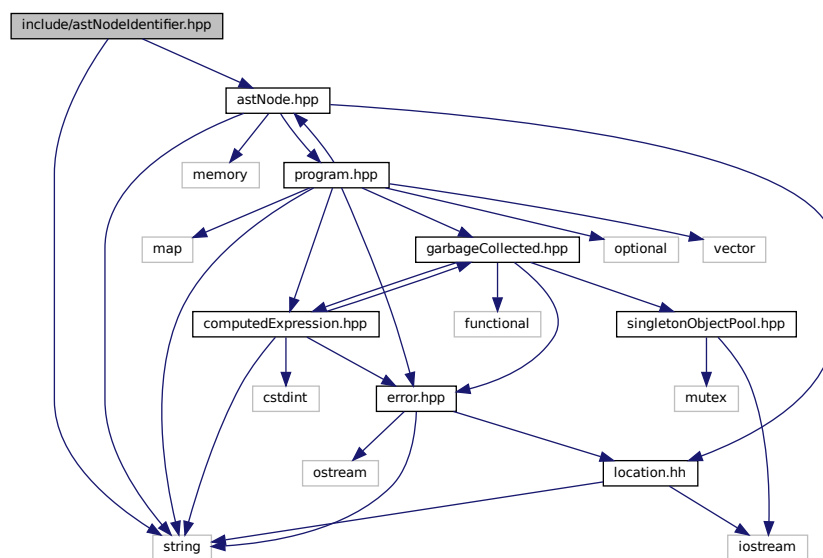
Declare the [Tang::AstNodeFloat](#) class.

6.9 include/astNodeIdentifier.hpp File Reference

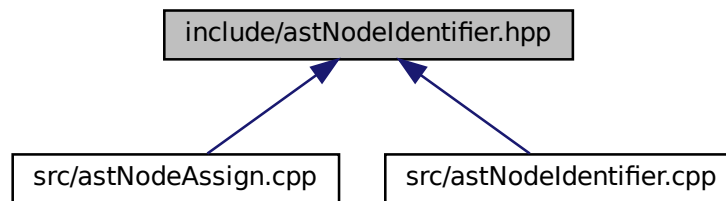
Declare the [Tang::AstNodeIdentifier](#) class.

```
#include <string>
#include "astNode.hpp"
```

Include dependency graph for astNodeIdentifier.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::AstNodeIdentifier](#)
An [AstNode](#) that represents an identifier.

6.9.1 Detailed Description

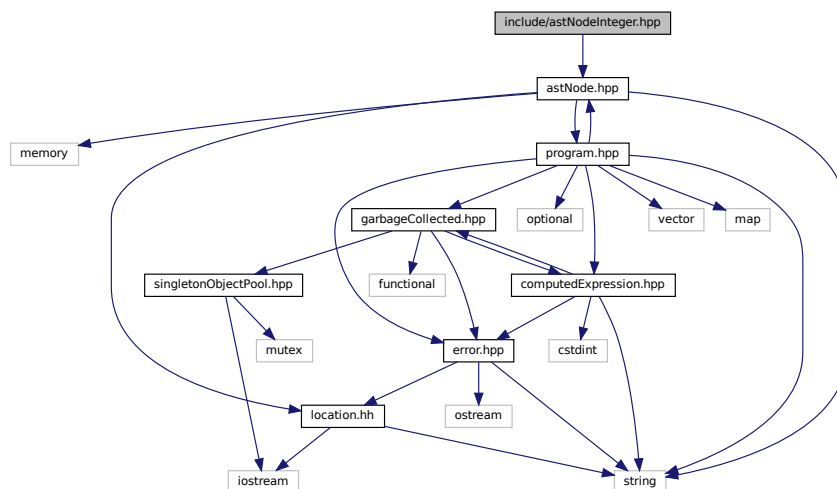
Declare the [Tang::AstNodeIdentifier](#) class.

6.10 include/astNodeInteger.hpp File Reference

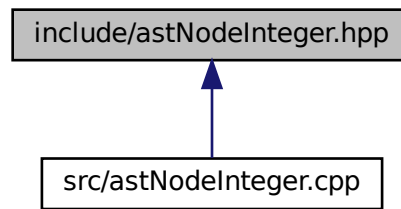
Declare the [Tang::AstNodeInteger](#) class.

```
#include "astNode.hpp"
```

Include dependency graph for `astNodeInteger.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::AstNodeInteger](#)
An [AstNode](#) that represents an integer literal.

6.10.1 Detailed Description

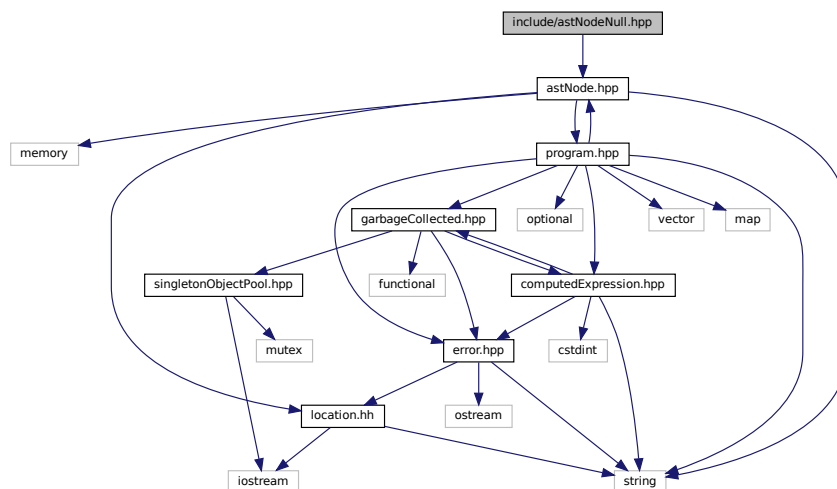
Declare the [Tang::AstNodeInteger](#) class.

6.11 include/astNodeNull.hpp File Reference

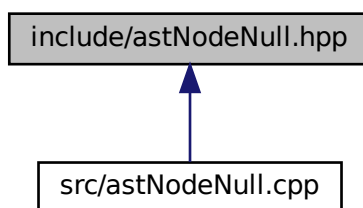
Declare the [Tang::AstNodeNull](#) class.

```
#include "astNode.hpp"
```

Include dependency graph for astNodeNull.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::AstNodeNull](#)
An [AstNode](#) that represents a `NULL` value.

6.11.1 Detailed Description

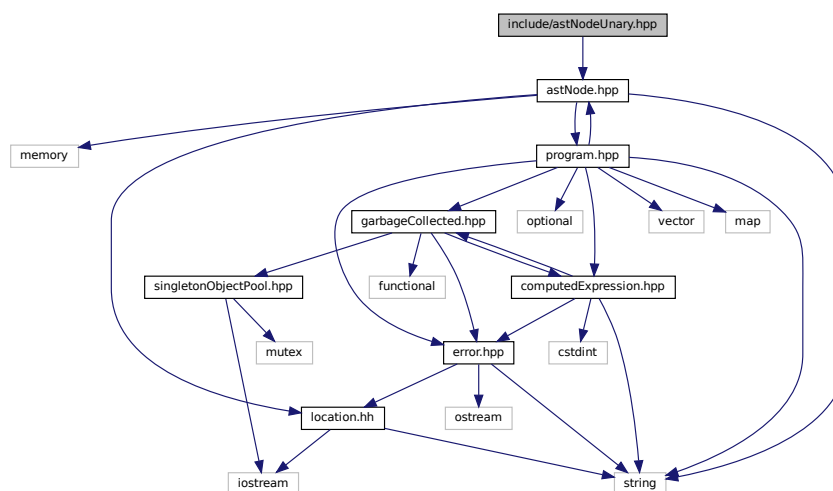
Declare the [Tang::AstNodeNull](#) class.

6.12 include/astNodeUnary.hpp File Reference

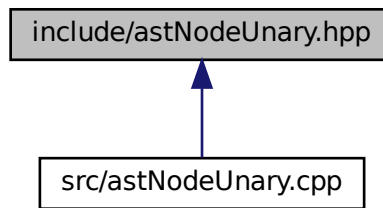
Declare the [Tang::AstNodeUnary](#) class.

```
#include "astNode.hpp"
```

Include dependency graph for `astNodeUnary.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::AstNodeUnary](#)
An [AstNode](#) that represents a unary negation.

6.12.1 Detailed Description

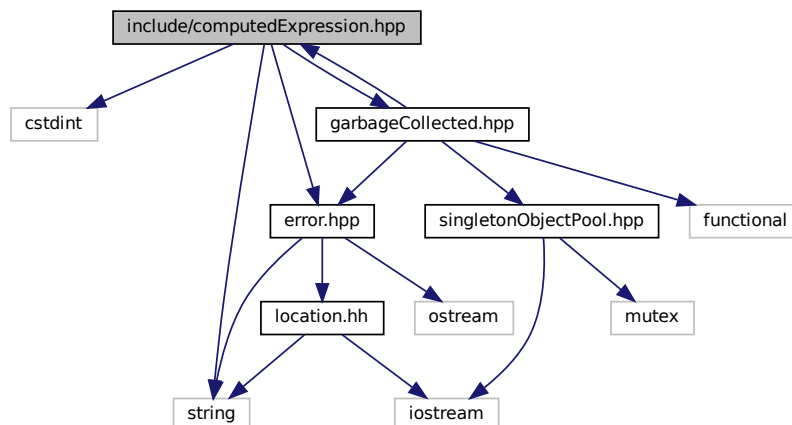
Declare the [Tang::AstNodeUnary](#) class.

6.13 include/computedExpression.hpp File Reference

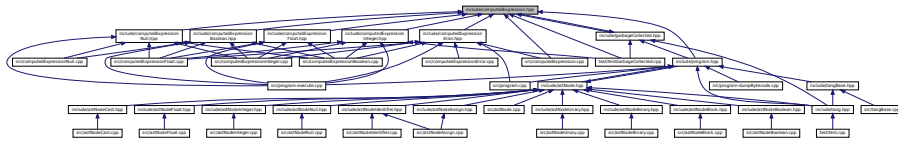
Declare the [Tang::ComputedExpression](#) base class.

```
#include <stdint>
#include <string>
#include "garbageCollected.hpp"
#include "error.hpp"
```

Include dependency graph for computedExpression.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::ComputedExpression](#)
Represents the result of a computation that has been executed.

6.13.1 Detailed Description

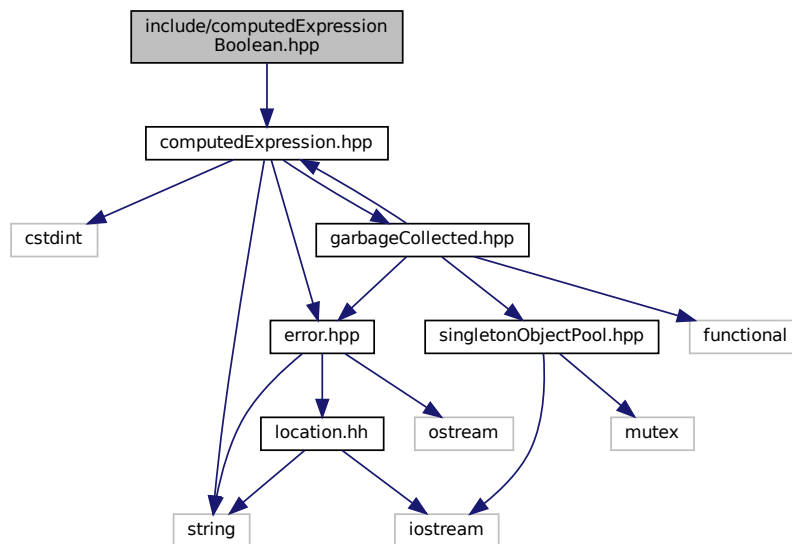
Declare the [Tang::ComputedExpression](#) base class.

6.14 include/computedExpressionBoolean.hpp File Reference

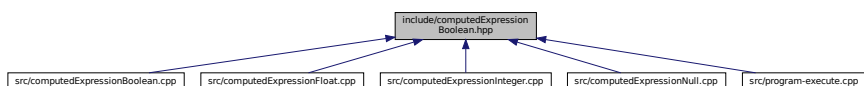
Declare the [Tang::ComputedExpressionBoolean](#) class.

```
#include "computedExpression.hpp"
```

Include dependency graph for computedExpressionBoolean.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::ComputedExpressionBoolean](#)
Represents an Boolean that is the result of a computation.

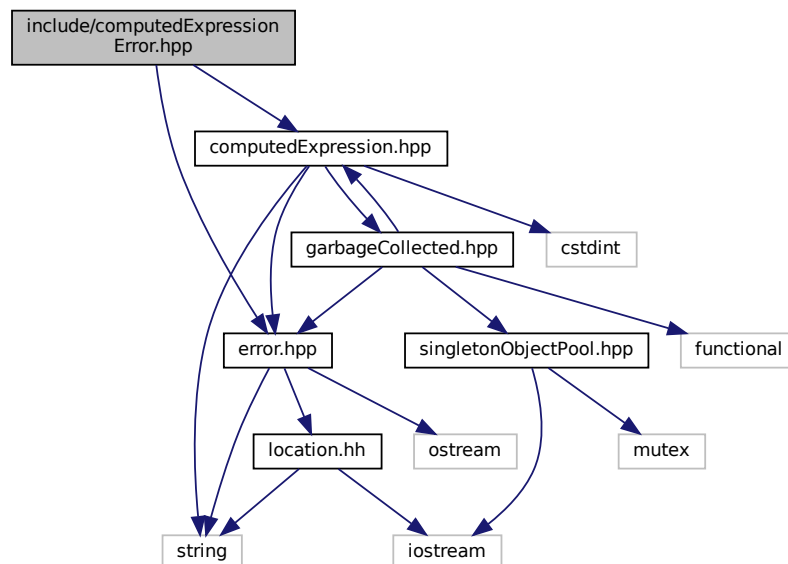
6.14.1 Detailed Description

Declare the [Tang::ComputedExpressionBoolean](#) class.

6.15 include/computedExpressionError.hpp File Reference

Declare the [Tang::ComputedExpressionError](#) class.

```
#include "computedExpression.hpp"
#include "error.hpp"
Include dependency graph for computedExpressionError.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::ComputedExpressionError](#)
Represents a Runtime Error.

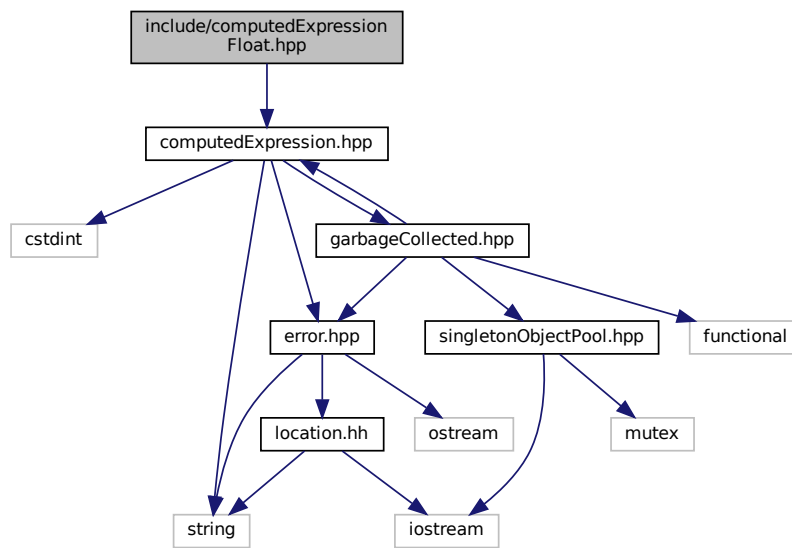
6.15.1 Detailed Description

Declare the [Tang::ComputedExpressionError](#) class.

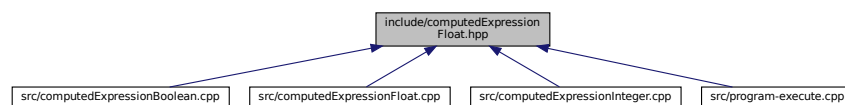
6.16 include/computedExpressionFloat.hpp File Reference

Declare the [Tang::ComputedExpressionFloat](#) class.

```
#include "computedExpression.hpp"
Include dependency graph for computedExpressionFloat.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::ComputedExpressionFloat](#)
Represents a Float that is the result of a computation.

6.16.1 Detailed Description

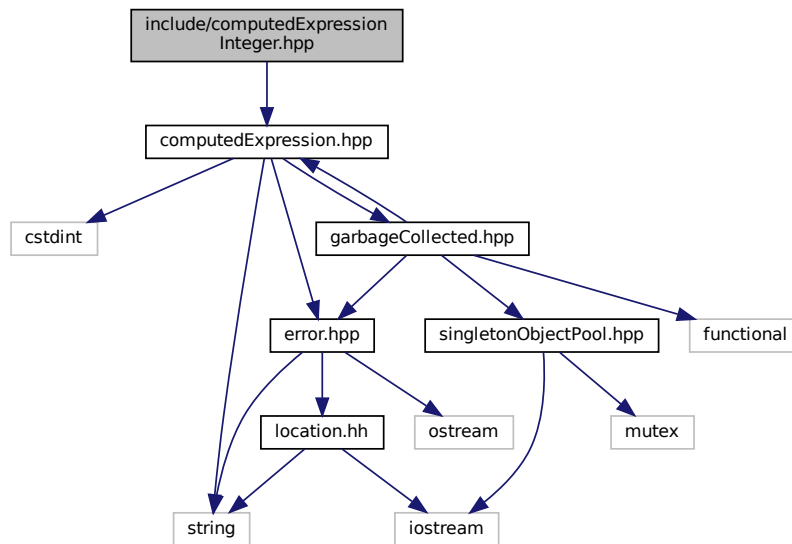
Declare the [Tang::ComputedExpressionFloat](#) class.

6.17 include/computedExpressionInteger.hpp File Reference

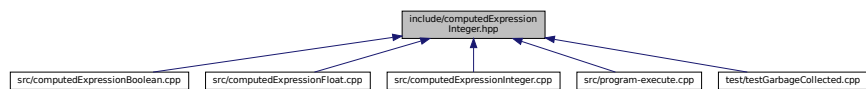
Declare the [Tang::ComputedExpressionInteger](#) class.

```
#include "computedExpression.hpp"
```

Include dependency graph for computedExpressionInteger.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::ComputedExpressionInteger](#)
Represents an Integer that is the result of a computation.

6.17.1 Detailed Description

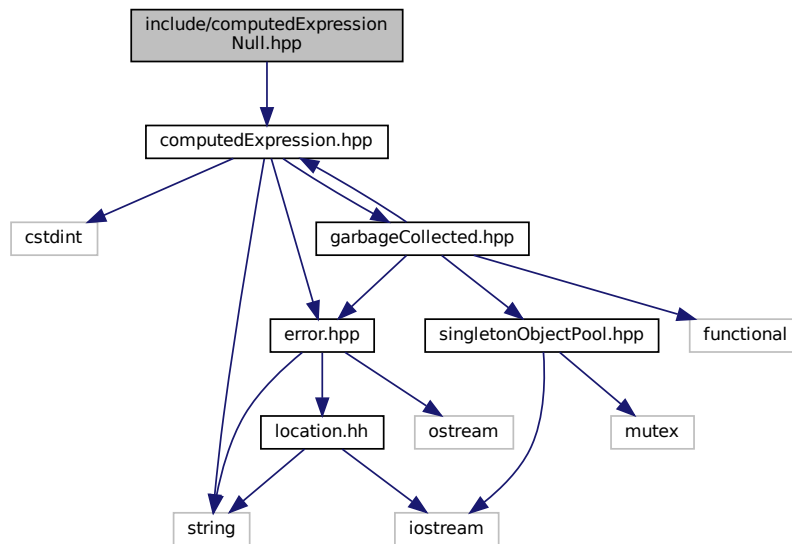
Declare the [Tang::ComputedExpressionInteger](#) class.

6.18 include/computedExpressionNull.hpp File Reference

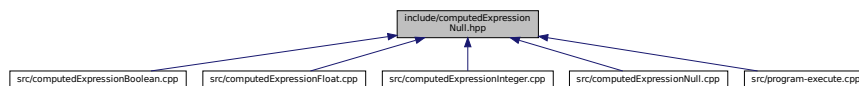
Declare the [Tang::ComputedExpressionNull](#) class.

```
#include "computedExpression.hpp"
```

Include dependency graph for computedExpressionNull.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::ComputedExpressionNull](#)
Represents an Null that is the result of a computation.

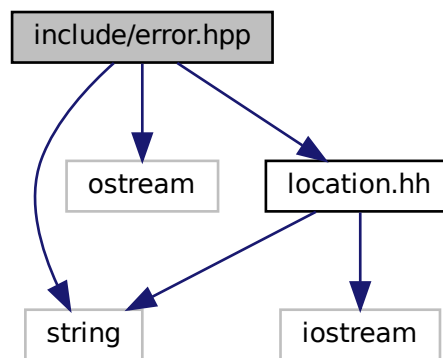
6.18.1 Detailed Description

Declare the [Tang::ComputedExpressionNull](#) class.

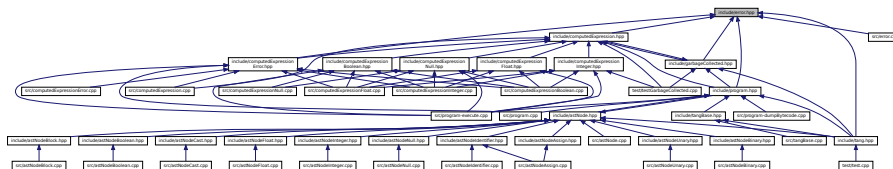
6.19 include/error.hpp File Reference

Declare the [Tang::Error](#) class used to describe syntax and runtime errors.

```
#include <string>
#include <ostream>
#include "location.hh"
Include dependency graph for error.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::Error](#)

The [Error](#) class is used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error.

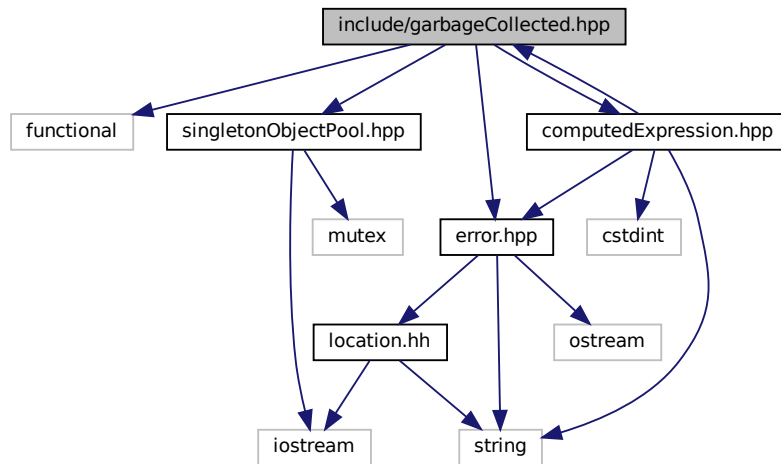
6.19.1 Detailed Description

Declare the [Tang::Error](#) class used to describe syntax and runtime errors.

6.20 include/garbageCollected.hpp File Reference

Declare the [Tang::GarbageCollected](#) class.

```
#include <functional>
#include "singletonObjectPool.hpp"
#include "computedExpression.hpp"
#include "error.hpp"
Include dependency graph for garbageCollected.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::GarbageCollected](#)

A container that acts as a resource-counting garbage collector for the specified type.

6.20.1 Detailed Description

Declare the [Tang::GarbageCollected](#) class.

6.21 include/macros.hpp File Reference

Contains generic macros.

6.22.1 Detailed Description

Declare the Opcodes used in the Bytecode representation of a program.

6.22.2 Enumeration Type Documentation

6.22.2.1 Opcode

```
enum Tang::Opcode [strong]
```

Enumerator

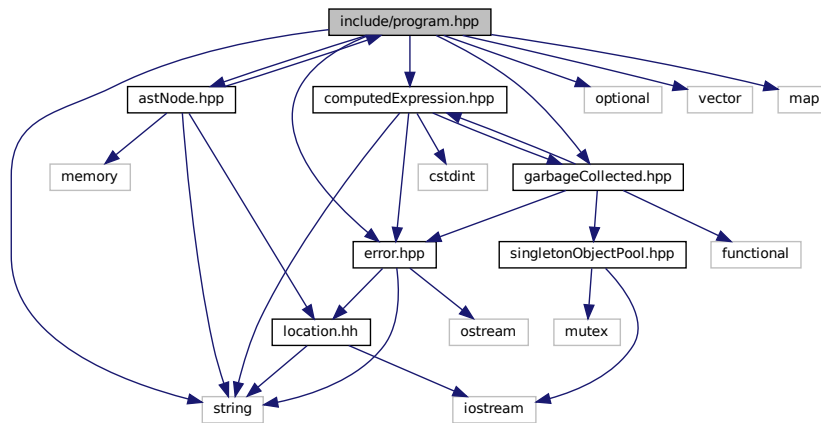
POP	Pop a val.
PEEK	Stack # (from fp): push val from stack #.
POKE	Stack # (from fp): Copy a val, store @ stack #.
NULLVAL	Push a null onto the stack.
INTEGER	Push an integer onto the stack.
FLOAT	Push a floating point number onto the stack.
BOOLEAN	Push a boolean onto the stack.
ADD	Pop rhs, pop lhs, push lhs + rhs.
SUBTRACT	Pop rhs, pop lhs, push lhs - rhs.
MULTIPLY	Pop rhs, pop lhs, push lhs * rhs.
DIVIDE	Pop rhs, pop lhs, push lhs / rhs.
MODULO	Pop rhs, pop lhs, push lhs % rhs.
NEGATIVE	Pop val, push negative val.
NOT	Pop val, push logical not of val.
LT	Pop rhs, pop lhs, push lhs < rhs.
LTE	Pop rhs, pop lhs, push lhs <= rhs.
GT	Pop rhs, pop lhs, push lhs > rhs.
GTE	Pop rhs, pop lhs, push lhs >= rhs.
EQ	Pop rhs, pop lhs, push lhs == rhs.
NEQ	Pop rhs, pop lhs, push lhs != rhs.
CASTINTEGER	Pop a val, typecast to int, push.
CASTFLOAT	Pop a val, typecast to float, push.
CASTBOOLEAN	Pop a val, typecast to boolean, push.

6.23 include/program.hpp File Reference

Declare the [Tang::Program](#) class used to compile and execute source code.

```
#include <string>
#include <optional>
#include <vector>
```

```
#include <map>
#include "astNode.hpp"
#include "error.hpp"
#include "computedExpression.hpp"
#include "garbageCollected.hpp"
Include dependency graph for program.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::Program](#)
Represents a compiled script or template that may be executed.

Typedefs

- using [Tang::Bytecode](#) = `std::vector< uint64_t >`
Contains the Opcodes of a compiled program.

6.23.1 Detailed Description

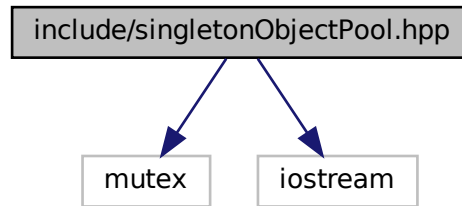
Declare the [Tang::Program](#) class used to compile and execute source code.

6.24 include/singletonObjectPool.hpp File Reference

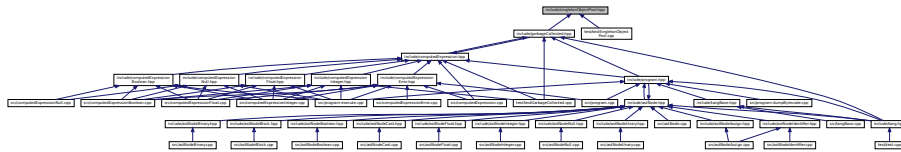
Declare the [Tang::SingletonObjectPool](#) class.

```
#include <mutex>
#include <iostream>
```

Include dependency graph for singletonObjectPool.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::SingletonObjectPool< T >](#)
A thread-safe, singleton object pool of the designated type.

Macros

- #define [GROW](#) 1024
The threshold size to use when allocating blocks of data, measured in the number of instances of the object type.

6.24.1 Detailed Description

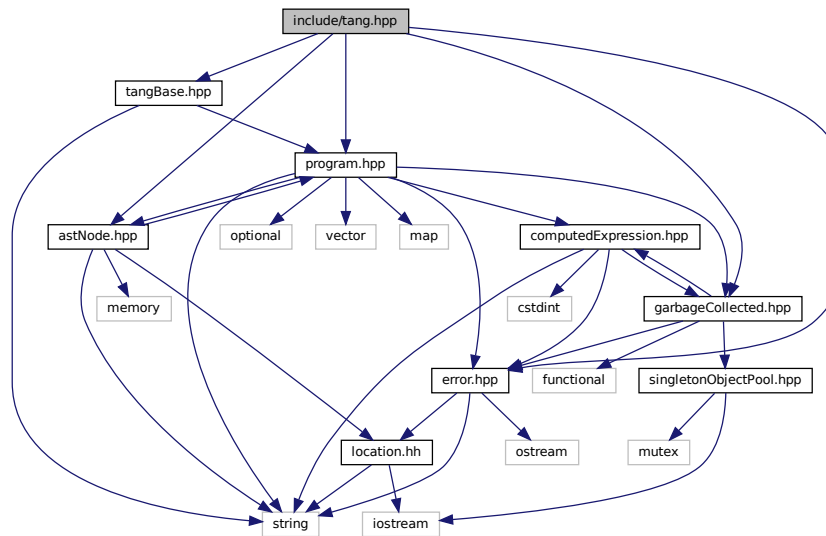
Declare the [Tang::SingletonObjectPool](#) class.

6.25 include/tang.hpp File Reference

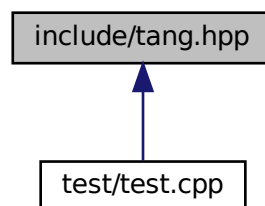
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

```
#include "tangBase.hpp"
#include "astNode.hpp"
#include "error.hpp"
#include "garbageCollected.hpp"
#include "program.hpp"
```

Include dependency graph for tang.hpp:



This graph shows which files directly or indirectly include this file:



6.25.1 Detailed Description

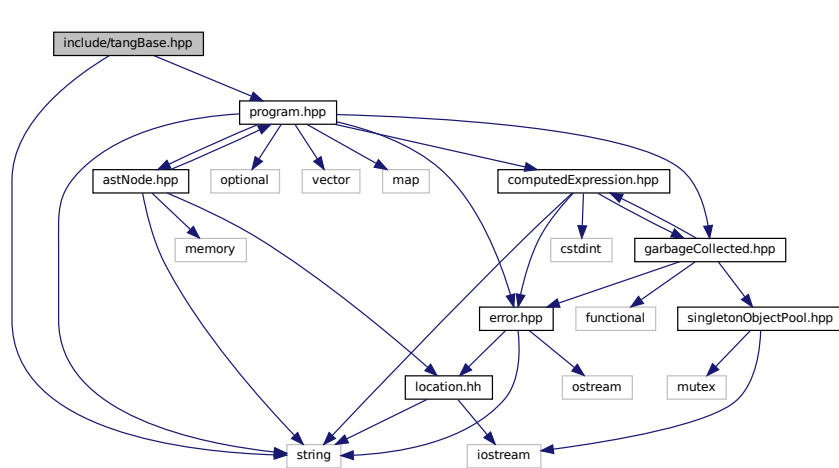
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

6.26 include/tangBase.hpp File Reference

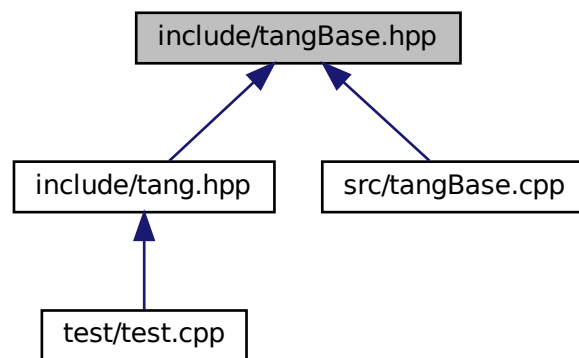
Declare the [Tang::TangBase](#) class used to interact with Tang.

```
#include <string>
#include "program.hpp"
```

Include dependency graph for tangBase.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::TangBase](#)

The base class for the Tang programming language.

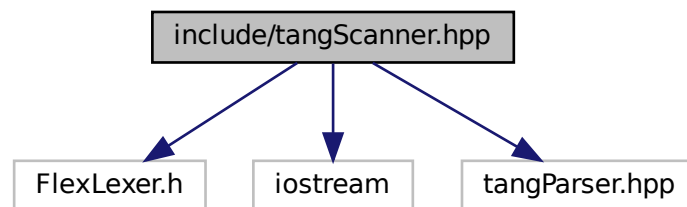
6.26.1 Detailed Description

Declare the [Tang::TangBase](#) class used to interact with Tang.

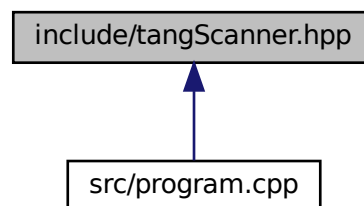
6.27 include/tangScanner.hpp File Reference

Declare the [Tang::TangScanner](#) used to tokenize a Tang script.

```
#include <FlexLexer.h>
#include <iostream>
#include "tangParser.hpp"
Include dependency graph for tangScanner.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Tang::TangScanner](#)

The Flex lexer class for the main Tang language.

Macros

- `#define yyFlexLexer TangTangFlexLexer`
- `#define YY_DECL Tang::TangParser::symbol_type Tang::TangScanner::get_next_token()`

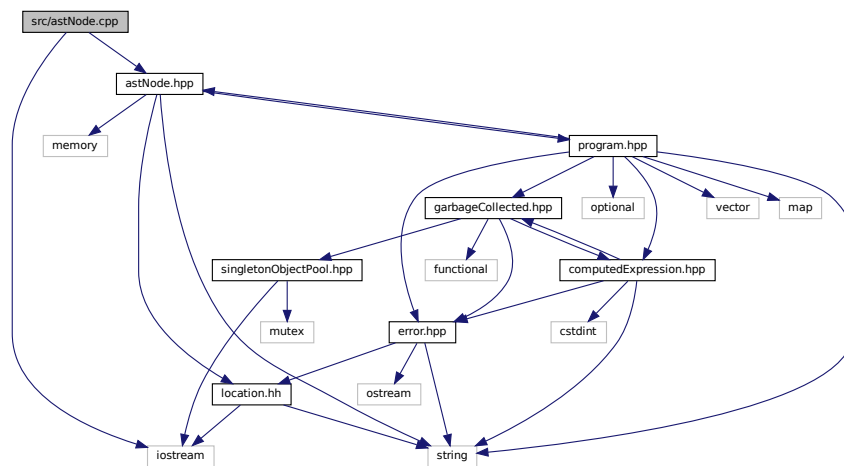
6.27.1 Detailed Description

Declare the [Tang::TangScanner](#) used to tokenize a Tang script.

6.28 src/astNode.cpp File Reference

Define the [Tang::AstNode](#) class.

```
#include <iostream>
#include "astNode.hpp"
Include dependency graph for astNode.cpp:
```



6.28.1 Detailed Description

Define the [Tang::AstNode](#) class.

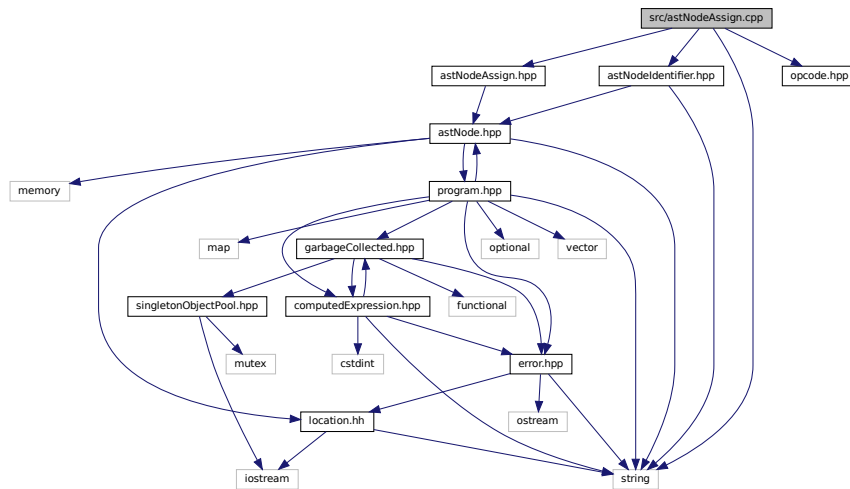
6.29 src/astNodeAssign.cpp File Reference

Define the [Tang::AstNodeAssign](#) class.

```
#include <string>
#include "astNodeAssign.hpp"
#include "astNodeIdentifier.hpp"
```

```
#include "opcode.hpp"
```

Include dependency graph for astNodeAssign.cpp:



6.29.1 Detailed Description

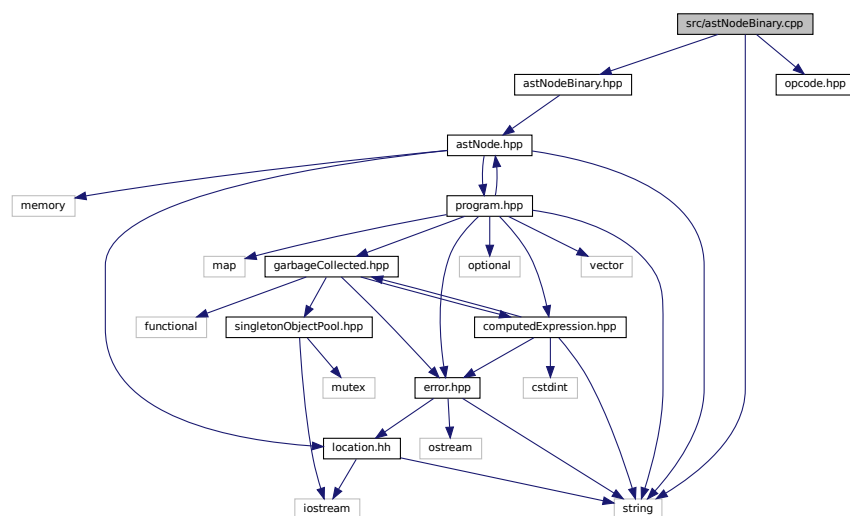
Define the [Tang::AstNodeAssign](#) class.

6.30 src/astNodeBinary.cpp File Reference

Define the [Tang::AstNodeBinary](#) class.

```
#include <string>
#include "astNodeBinary.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeBinary.cpp:



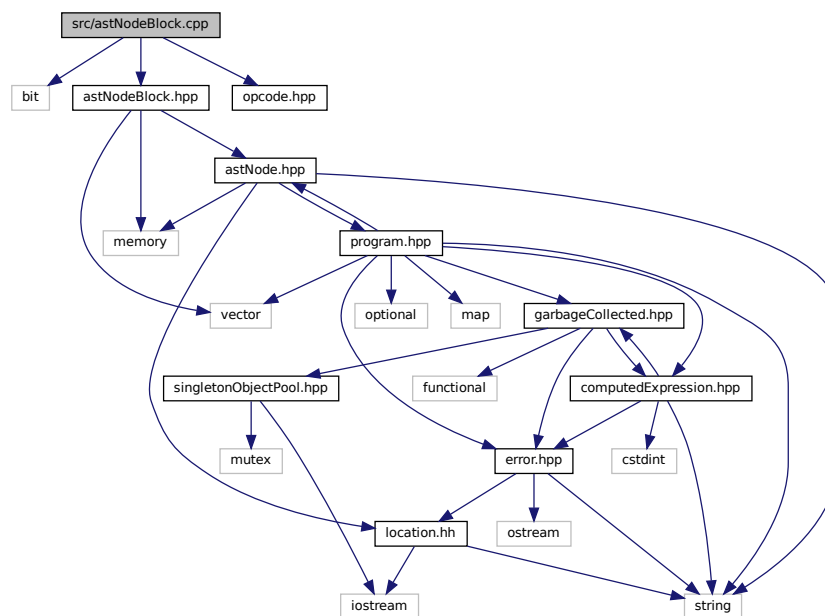
6.30.1 Detailed Description

Define the [Tang::AstNodeBinary](#) class.

6.31 src/astNodeBlock.cpp File Reference

Define the [Tang::AstNodeBlock](#) class.

```
#include <bit>
#include "astNodeBlock.hpp"
#include "opcode.hpp"
Include dependency graph for astNodeBlock.cpp:
```



6.31.1 Detailed Description

Define the [Tang::AstNodeBlock](#) class.

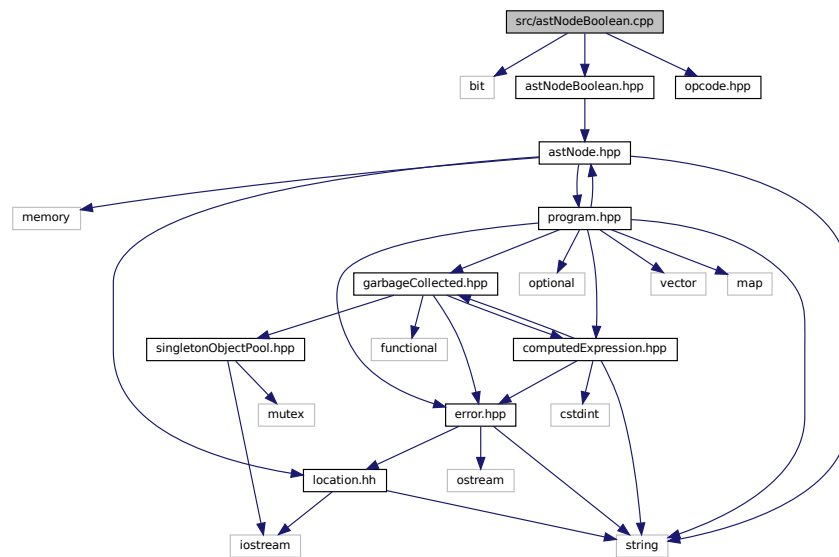
6.32 src/astNodeBoolean.cpp File Reference

Define the [Tang::AstNodeBoolean](#) class.

```
#include <bit>
#include "astNodeBoolean.hpp"
```

```
#include "opcode.hpp"
```

Include dependency graph for astNodeBoolean.cpp:



6.32.1 Detailed Description

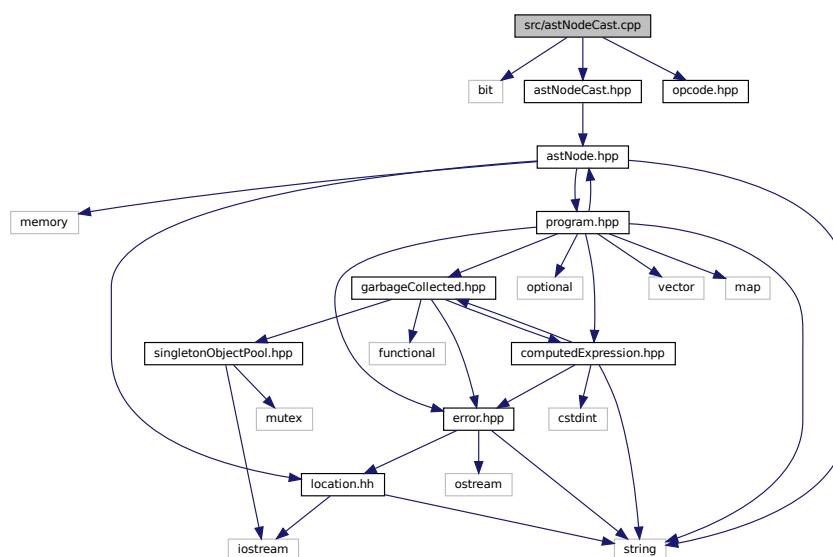
Define the [Tang::AstNodeBoolean](#) class.

6.33 src/astNodeCast.cpp File Reference

Define the [Tang::AstNodeCast](#) class.

```
#include <bit>
#include "astNodeCast.hpp"
#include "opcode.hpp"
```


Include dependency graph for astNodeCast.cpp:



6.33.1 Detailed Description

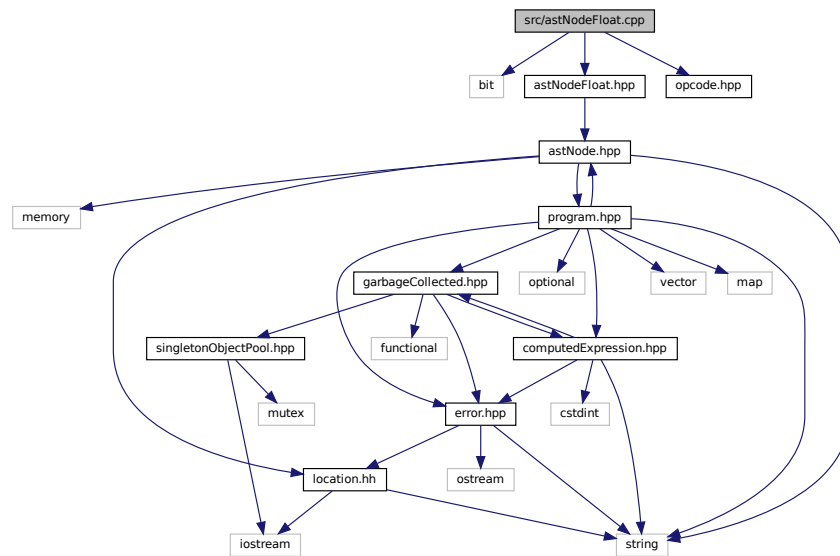
Define the [Tang::AstNodeCast](#) class.

6.34 src/astNodeFloat.cpp File Reference

Define the [Tang::AstNodeFloat](#) class.

```
#include <bit>
#include "astNodeFloat.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeFloat.cpp:



6.34.1 Detailed Description

Define the [Tang::AstNodeFloat](#) class.

6.35 src/astNodeIdentifier.cpp File Reference

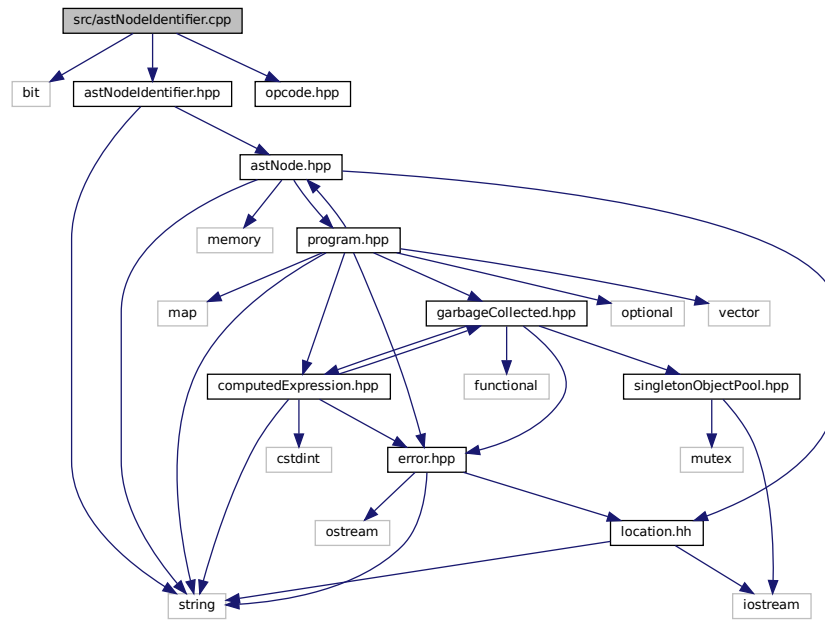
Define the [Tang::AstNodeIdentifier](#) class.

```

#include <bit>
#include "astNodeIdentifier.hpp"
#include "opcode.hpp"

```

Include dependency graph for astNodeIdentifier.cpp:



6.35.1 Detailed Description

Define the [Tang::AstNodeIdentifier](#) class.

6.36 src/astNodeInteger.cpp File Reference

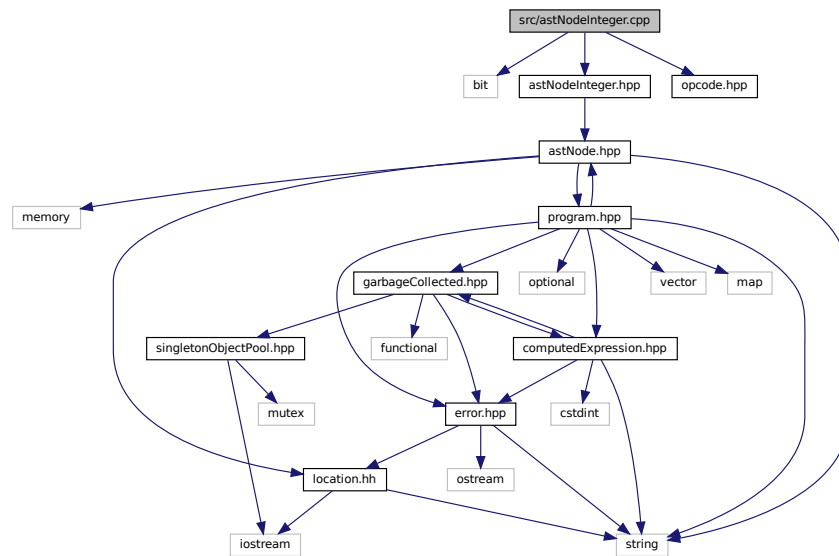
Define the [Tang::AstNodeInteger](#) class.

```

#include <bit>
#include "astNodeInteger.hpp"
#include "opcode.hpp"

```

Include dependency graph for `astNodeInteger.cpp`:



6.36.1 Detailed Description

Define the [Tang::AstNodeInteger](#) class.

6.37 src/astNodeNull.cpp File Reference

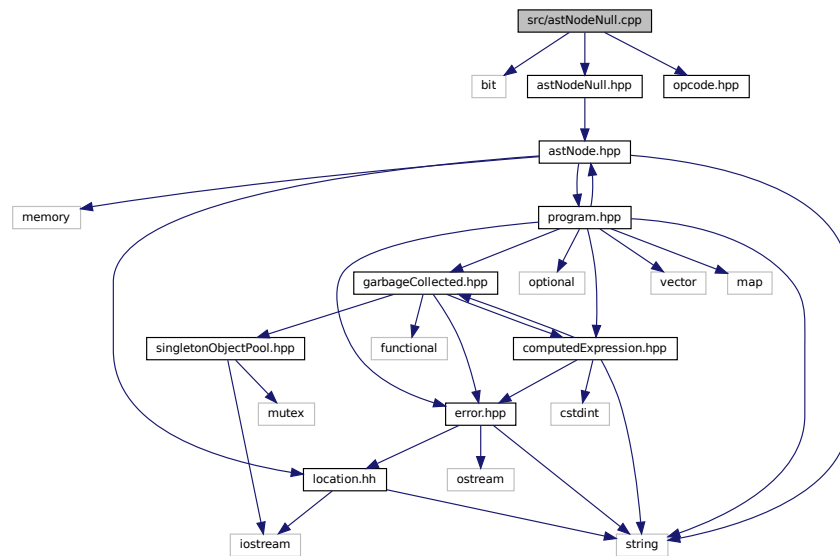
Define the [Tang::AstNodeNull](#) class.

```

#include <bit>
#include "astNodeNull.hpp"
#include "opcode.hpp"

```

Include dependency graph for astNodeNull.cpp:



6.37.1 Detailed Description

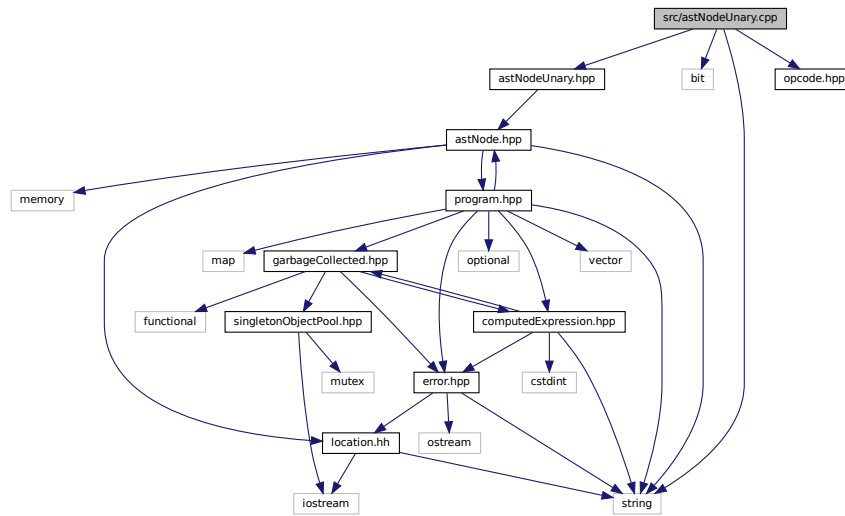
Define the [Tang::AstNodeNull](#) class.

6.38 src/astNodeUnary.cpp File Reference

Define the [Tang::AstNodeUnary](#) class.

```
#include <string>
#include <bit>
#include "astNodeUnary.hpp"
#include "opcode.hpp"
```

Include dependency graph for `astNodeUnary.cpp`:



6.38.1 Detailed Description

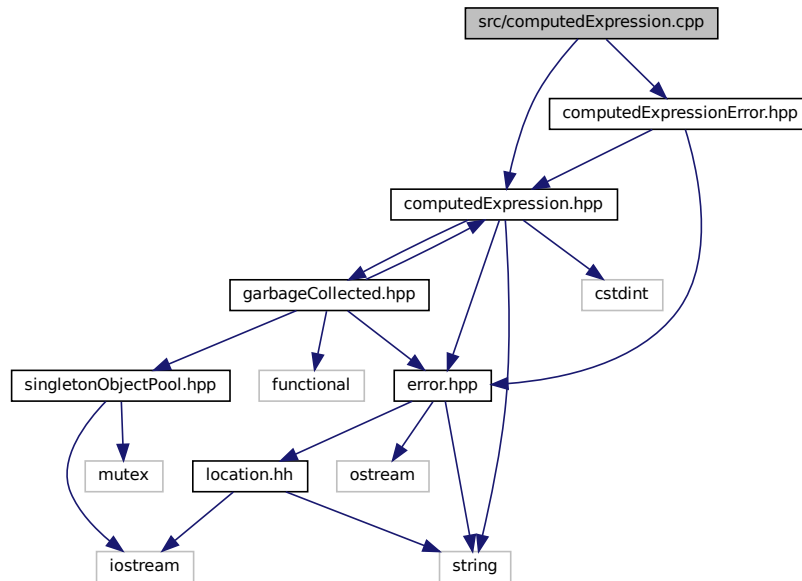
Define the [Tang::AstNodeUnary](#) class.

6.39 src/computedExpression.cpp File Reference

Define the [Tang::ComputedExpression](#) class.

```
#include "computedExpression.hpp"
#include "computedExpressionError.hpp"
```

Include dependency graph for computedExpression.cpp:



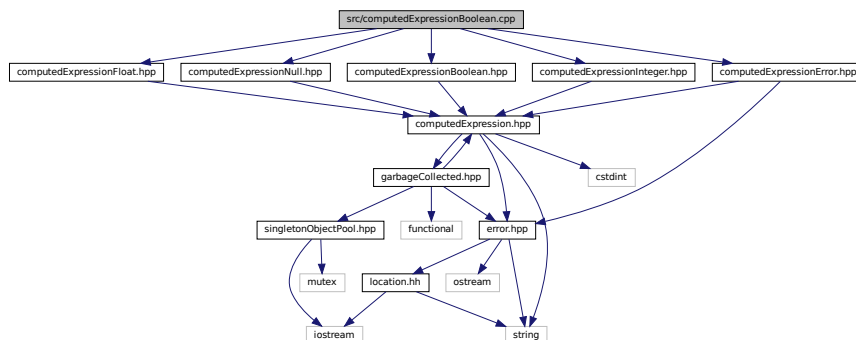
6.39.1 Detailed Description

Define the [Tang::ComputedExpression](#) class.

6.40 src/computedExpressionBoolean.cpp File Reference

Define the [Tang::ComputedExpressionBoolean](#) class.

```
#include "computedExpressionNull.hpp"
#include "computedExpressionBoolean.hpp"
#include "computedExpressionInteger.hpp"
#include "computedExpressionFloat.hpp"
#include "computedExpressionError.hpp"
Include dependency graph for computedExpressionBoolean.cpp:
```



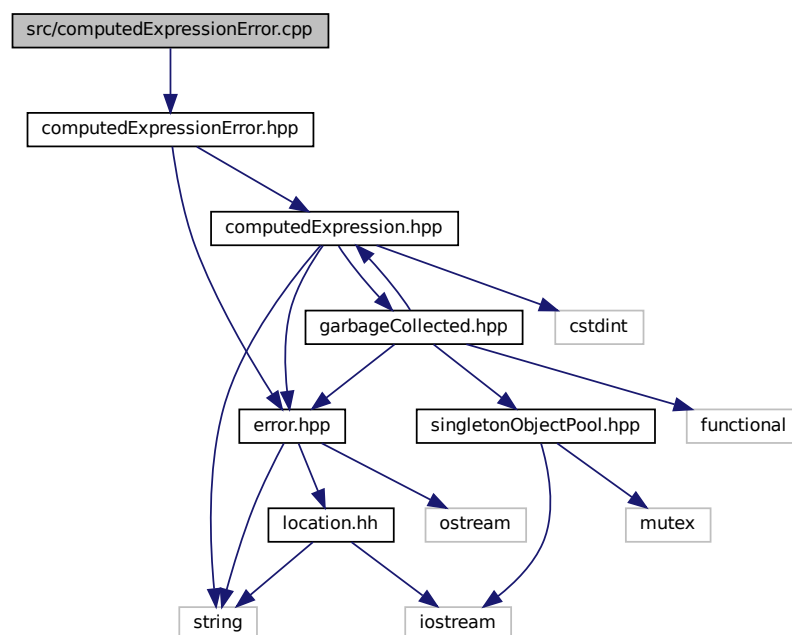
6.40.1 Detailed Description

Define the [Tang::ComputedExpressionBoolean](#) class.

6.41 src/computedExpressionError.cpp File Reference

Define the [Tang::ComputedExpressionError](#) class.

```
#include "computedExpressionError.hpp"
Include dependency graph for computedExpressionError.cpp:
```



6.41.1 Detailed Description

Define the [Tang::ComputedExpressionError](#) class.

6.42 src/computedExpressionFloat.cpp File Reference

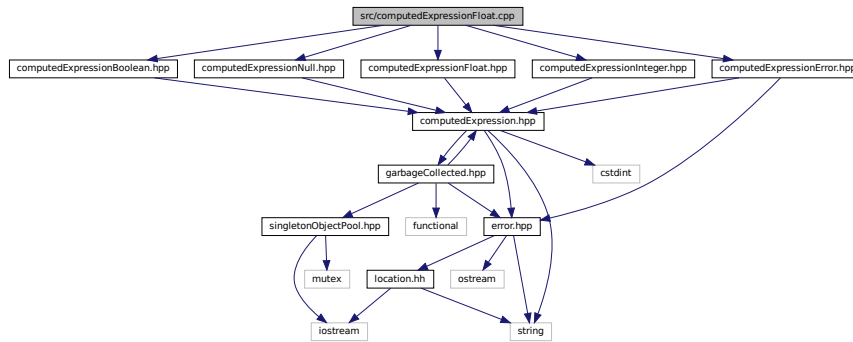
Define the [Tang::ComputedExpressionFloat](#) class.

```
#include "computedExpressionNull.hpp"
#include "computedExpressionFloat.hpp"
#include "computedExpressionInteger.hpp"
#include "computedExpressionBoolean.hpp"
```



```
#include "computedExpressionError.hpp"
```

Include dependency graph for computedExpressionFloat.cpp:



6.42.1 Detailed Description

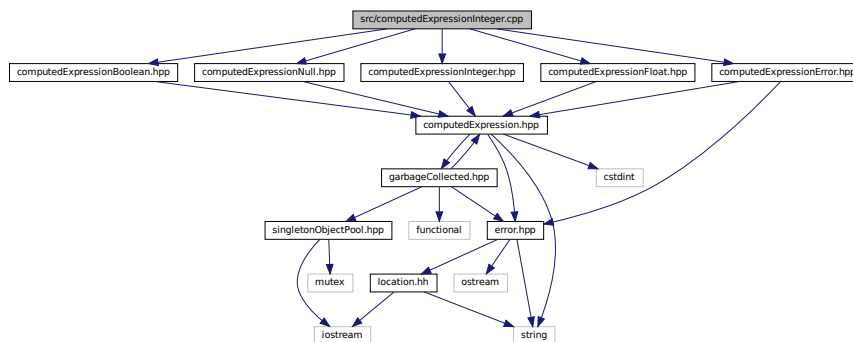
Define the [Tang::ComputedExpressionFloat](#) class.

6.43 src/computedExpressionInteger.cpp File Reference

Define the [Tang::ComputedExpressionInteger](#) class.

```
#include "computedExpressionNull.hpp"
#include "computedExpressionInteger.hpp"
#include "computedExpressionFloat.hpp"
#include "computedExpressionBoolean.hpp"
#include "computedExpressionError.hpp"
```

Include dependency graph for computedExpressionInteger.cpp:



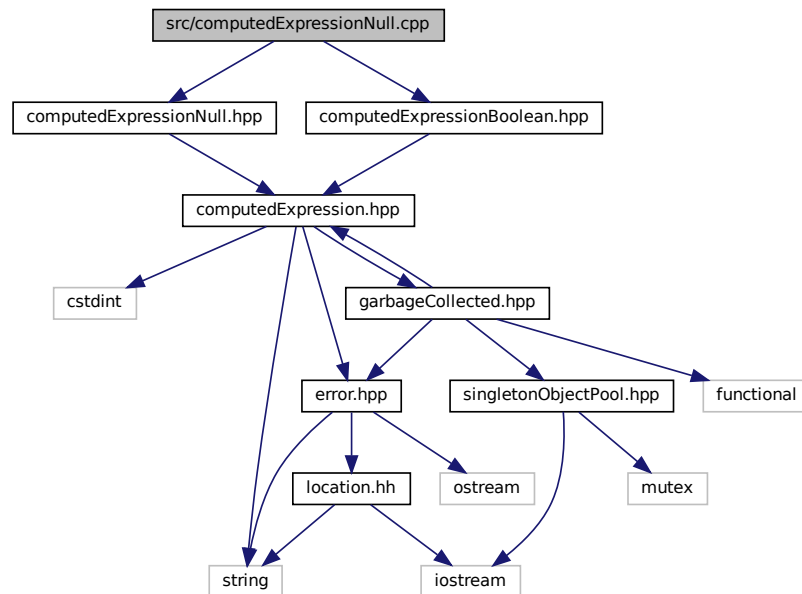
6.43.1 Detailed Description

Define the [Tang::ComputedExpressionInteger](#) class.

6.44 src/computedExpressionNull.cpp File Reference

Define the [Tang::ComputedExpressionNull](#) class.

```
#include "computedExpressionNull.hpp"
#include "computedExpressionBoolean.hpp"
Include dependency graph for computedExpressionNull.cpp:
```



6.44.1 Detailed Description

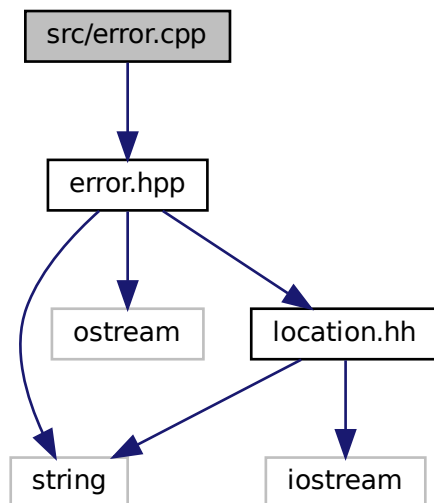
Define the [Tang::ComputedExpressionNull](#) class.

6.45 src/error.cpp File Reference

Define the [Tang::Error](#) class.

```
#include "error.hpp"
```

Include dependency graph for error.cpp:



Functions

- `std::ostream & Tang::operator<< (std::ostream &out, const Error &error)`

6.45.1 Detailed Description

Define the `Tang::Error` class.

6.45.2 Function Documentation

6.45.2.1 operator<<()

```
std::ostream& Tang::operator<< (
    std::ostream & out,
    const Error & error )
```

Parameters

<i>out</i>	The output stream.
<i>error</i>	The Error object.

Returns

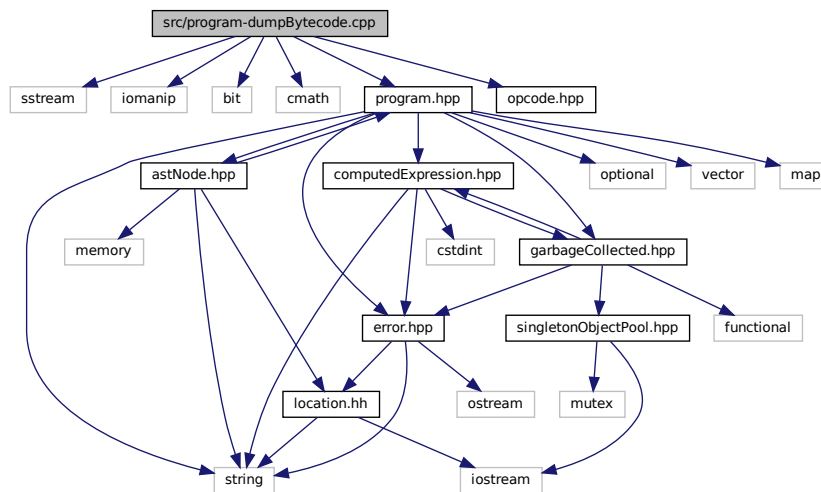
The output stream.

6.46 src/program-dumpBytecode.cpp File Reference

Define the [Tang::Program::dumpBytecode](#) method.

```
#include <sstream>
#include <iomanip>
#include <bit>
#include <cmath>
#include "program.hpp"
#include "opcode.hpp"
```

Include dependency graph for program-dumpBytecode.cpp:



Macros

- `#define DUMPPROGRAMCHECK(x)`
Verify the size of the Bytecode vector so that it may be safely accessed.

6.46.1 Detailed Description

Define the [Tang::Program::dumpBytecode](#) method.

6.46.2 Macro Definition Documentation

6.46.2.1 DUMPPROGRAMCHECK

```
#define DUMPPROGRAMCHECK(
    x )
```

Value:

```
if (this->bytecode.size() < (pc + (x))) \
    return out.str() + "Error: Opcode truncated\n"
```

Verify the size of the Bytecode vector so that it may be safely accessed.

If the vector is not large enough, an error message is appended to the output string and no further opcodes are printed.

Parameters

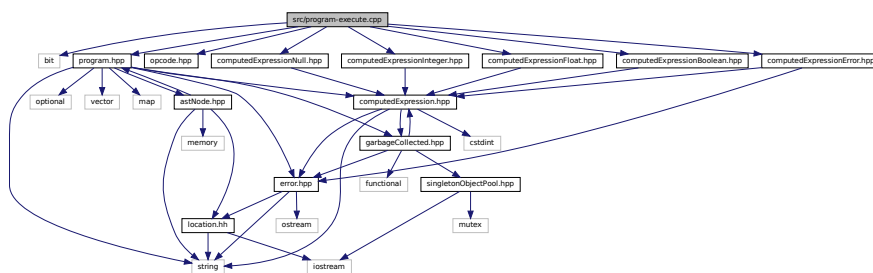
x	The number of additional vector entries that should exist.
---	--

6.47 src/program-execute.cpp File Reference

Define the [Tang::Program::execute](#) method.

```
#include <bit>
#include "program.hpp"
#include "opcode.hpp"
#include "computedExpressionError.hpp"
#include "computedExpressionNull.hpp"
#include "computedExpressionInteger.hpp"
#include "computedExpressionFloat.hpp"
#include "computedExpressionBoolean.hpp"
```

Include dependency graph for program-execute.cpp:

**Macros**

- `#define EXECUTEPROGRAMCHECK(x)`
Verify the size of the Bytecode vector so that it may be safely accessed.
- `#define STACKCHECK(x)`
Verify the size of the stack vector so that it may be safely accessed.

6.47.1 Detailed Description

Define the [Tang::Program::execute](#) method.

6.47.2 Macro Definition Documentation

6.47.2.1 EXECUTEPROGRAMCHECK

```
#define EXECUTEPROGRAMCHECK(  
    x )
```

Value:

```
if (this->bytecode.size() < (pc + (x))) { \
    stack.push_back(GarbageCollected::make<ComputedExpressionError>(Error{"Opcode instruction  
truncated."})); \
    pc = this->bytecode.size(); \
    break; \
}
```

Verify the size of the Bytecode vector so that it may be safely accessed.

Parameters

x	The number of additional vector entries that should exist.
---	--

6.47.2.2 STACKCHECK

```
#define STACKCHECK(  
    x )
```

Value:

```
if (stack.size() < (fp + (x))) { \
    stack.push_back(GarbageCollected::make<ComputedExpressionError>(Error{"Insufficient stack depth."})); \
    pc = this->bytecode.size(); \
    break; \
}
```

Verify the size of the stack vector so that it may be safely accessed.

Parameters

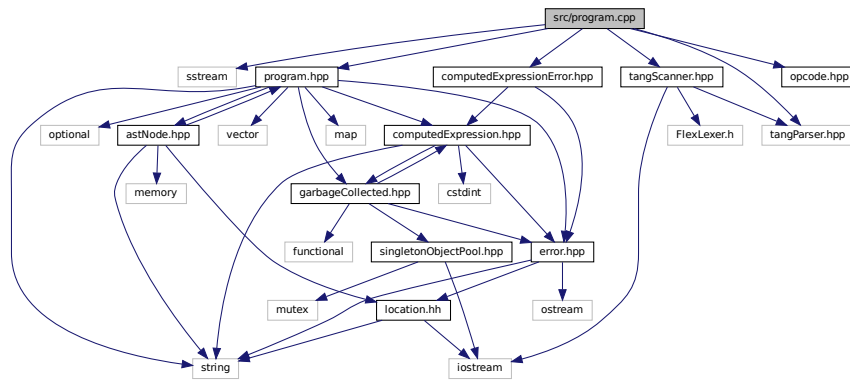
x	The number of entries that should exist in the stack.
---	---

6.48 src/program.cpp File Reference

Define the [Tang::Program](#) class.

```
#include <sstream>
#include "program.hpp"
#include "opcode.hpp"
#include "tangScanner.hpp"
#include "tangParser.hpp"
#include "computedExpressionError.hpp"
```

Include dependency graph for program.cpp:



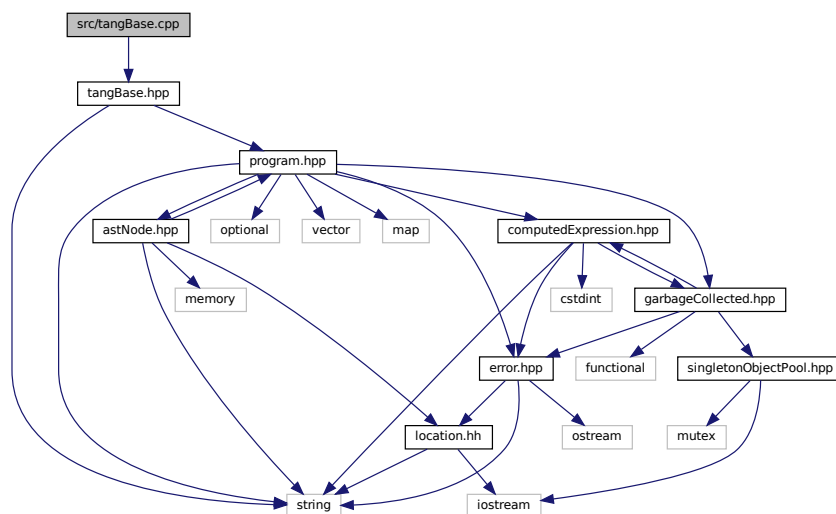
6.48.1 Detailed Description

Define the [Tang::Program](#) class.

6.49 src/tangBase.cpp File Reference

Define the [Tang::TangBase](#) class.

```
#include "tangBase.hpp"
Include dependency graph for tangBase.cpp:
```



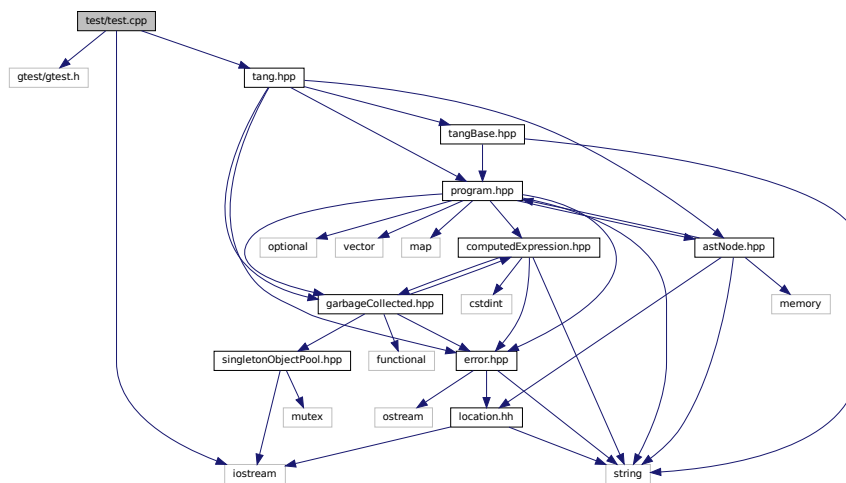
6.49.1 Detailed Description

Define the `Tang::TangBase` class.

6.50 test/test.cpp File Reference

Test the general language behaviors.

```
#include <gtest/gtest.h>
#include <iostream>
#include "tang.hpp"
Include dependency graph for test.cpp:
```



Functions

- **TEST** (Declare, Null)
- **TEST** (Declare, Integer)
- **TEST** (Declare, Float)
- **TEST** (Expression, Add)
- **TEST** (Expression, Subtract)
- **TEST** (Expression, Multiplication)
- **TEST** (Expression, Division)
- **TEST** (Expression, Modulo)
- **TEST** (Expression, UnaryMinus)
- **TEST** (Expression, Parentheses)
- **TEST** (Expression, TypeCast)
- **TEST** (Expression, Boolean)
- **TEST** (Expression, Not)
- **TEST** (Expression, LessThan)
- **TEST** (Expression, LessThanEqual)
- **TEST** (Expression, GreaterThan)
- **TEST** (Expression, GreaterThanEqual)
- **TEST** (Expression, Equal)
- **TEST** (Expression, NotEqual)
- **TEST** (CodeBlock, Statements)
- **TEST** (Assign, Identifier)
- **int main** (int argc, char **argv)

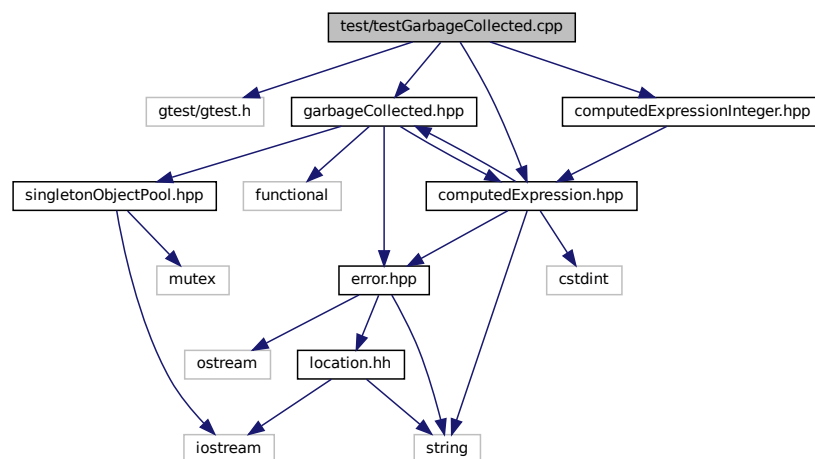
6.50.1 Detailed Description

Test the general language behaviors.

6.51 test/testGarbageCollected.cpp File Reference

Test the generic behavior of the [Tang::GarbageCollected](#) class.

```
#include <gtest/gtest.h>
#include "garbageCollected.hpp"
#include "computedExpression.hpp"
#include "computedExpressionInteger.hpp"
Include dependency graph for testGarbageCollected.cpp:
```



Functions

- **TEST** (Create, Access)
- **TEST** (RuleOfFive, CopyConstructor)
- **TEST** (Recycle, ObjectIsRecycled)
- **TEST** (Recycle, ObjectIsNotRecycled)
- int **main** (int argc, char **argv)

6.51.1 Detailed Description

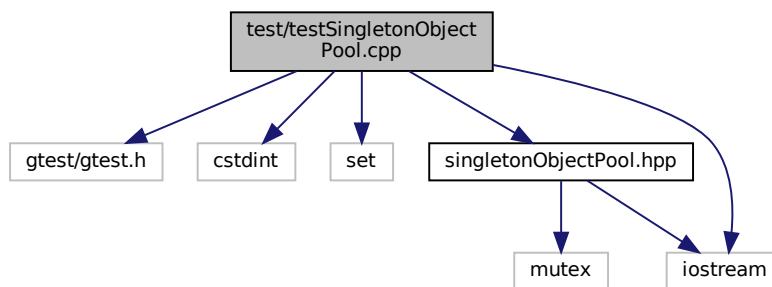
Test the generic behavior of the [Tang::GarbageCollected](#) class.

6.52 test/testSingletonObjectPool.cpp File Reference

Test the generic behavior of the [Tang::SingletonObjectPool](#) class.

```
#include <gtest/gtest.h>
#include <cstdlib>
#include <set>
#include "singletonObjectPool.hpp"
#include <iostream>
```

Include dependency graph for testSingletonObjectPool.cpp:



Functions

- **TEST** (Singleton, SameForSameType)
- **TEST** (Singleton, DifferentForDifferentTypes)
- **TEST** (Get, SuccessiveCallsProduceDifferentMemoryAddresses)
- **TEST** (Recycle, RecycledObjectIsReused)
- **TEST** (Get, SuccessiveCallsAreSequential)
- **TEST** (Get, KeepsGeneratingDifferentPointers)
- **TEST** (Recycle, WorksAfterLargeNumberOfAllocations)
- `int main` (`int argc`, `char **argv`)

6.52.1 Detailed Description

Test the generic behavior of the [Tang::SingletonObjectPool](#) class.

Index

- __add
 - Tang::ComputedExpression, [48](#)
 - Tang::ComputedExpressionBoolean, [57](#)
 - Tang::ComputedExpressionError, [66](#)
 - Tang::ComputedExpressionFloat, [75](#)
 - Tang::ComputedExpressionInteger, [84](#)
 - Tang::ComputedExpressionNull, [92](#)
- __boolean
 - Tang::ComputedExpression, [48](#)
 - Tang::ComputedExpressionBoolean, [57](#)
 - Tang::ComputedExpressionError, [66](#)
 - Tang::ComputedExpressionFloat, [75](#)
 - Tang::ComputedExpressionInteger, [84](#)
 - Tang::ComputedExpressionNull, [93](#)
- __divide
 - Tang::ComputedExpression, [49](#)
 - Tang::ComputedExpressionBoolean, [57](#)
 - Tang::ComputedExpressionError, [66](#)
 - Tang::ComputedExpressionFloat, [75](#)
 - Tang::ComputedExpressionInteger, [84](#)
 - Tang::ComputedExpressionNull, [93](#)
- __equal
 - Tang::ComputedExpression, [49](#)
 - Tang::ComputedExpressionBoolean, [58](#)
 - Tang::ComputedExpressionError, [67](#)
 - Tang::ComputedExpressionFloat, [76](#)
 - Tang::ComputedExpressionInteger, [85](#)
 - Tang::ComputedExpressionNull, [93](#)
- __float
 - Tang::ComputedExpression, [49](#)
 - Tang::ComputedExpressionBoolean, [58](#)
 - Tang::ComputedExpressionError, [67](#)
 - Tang::ComputedExpressionFloat, [76](#)
 - Tang::ComputedExpressionInteger, [85](#)
 - Tang::ComputedExpressionNull, [94](#)
- __integer
 - Tang::ComputedExpression, [50](#)
 - Tang::ComputedExpressionBoolean, [58](#)
 - Tang::ComputedExpressionError, [67](#)
 - Tang::ComputedExpressionFloat, [76](#)
 - Tang::ComputedExpressionInteger, [85](#)
 - Tang::ComputedExpressionNull, [94](#)
- __lessThan
 - Tang::ComputedExpression, [50](#)
 - Tang::ComputedExpressionBoolean, [59](#)
 - Tang::ComputedExpressionError, [68](#)
 - Tang::ComputedExpressionFloat, [77](#)
 - Tang::ComputedExpressionInteger, [86](#)
 - Tang::ComputedExpressionNull, [94](#)
- __modulo
 - Tang::ComputedExpression, [50](#)
 - Tang::ComputedExpressionBoolean, [59](#)
 - Tang::ComputedExpressionError, [68](#)
 - Tang::ComputedExpressionFloat, [77](#)
 - Tang::ComputedExpressionInteger, [86](#)
 - Tang::ComputedExpressionNull, [95](#)
- __multiply
 - Tang::ComputedExpression, [51](#)
 - Tang::ComputedExpressionBoolean, [59](#)
 - Tang::ComputedExpressionError, [68](#)
 - Tang::ComputedExpressionFloat, [77](#)
 - Tang::ComputedExpressionInteger, [86](#)
 - Tang::ComputedExpressionNull, [95](#)
- __negative
 - Tang::ComputedExpression, [51](#)
 - Tang::ComputedExpressionBoolean, [60](#)
 - Tang::ComputedExpressionError, [69](#)
 - Tang::ComputedExpressionFloat, [78](#)
 - Tang::ComputedExpressionInteger, [87](#)
 - Tang::ComputedExpressionNull, [95](#)
- __not
 - Tang::ComputedExpression, [51](#)
 - Tang::ComputedExpressionBoolean, [60](#)
 - Tang::ComputedExpressionError, [69](#)
 - Tang::ComputedExpressionFloat, [78](#)
 - Tang::ComputedExpressionInteger, [87](#)
 - Tang::ComputedExpressionNull, [96](#)
- __subtract
 - Tang::ComputedExpression, [52](#)
 - Tang::ComputedExpressionBoolean, [60](#)
 - Tang::ComputedExpressionError, [69](#)
 - Tang::ComputedExpressionFloat, [78](#)
 - Tang::ComputedExpressionInteger, [87](#)
 - Tang::ComputedExpressionNull, [96](#)
- ~GarbageCollected
 - Tang::GarbageCollected, [105](#)
- ADD
 - opcode.hpp, [153](#)
- Add
 - Tang::AstNodeBinary, [20](#)
- addBytecode
 - Tang::Program, [123](#)
- AstNode
 - Tang::AstNode, [14](#)
- AstNodeAssign
 - Tang::AstNodeAssign, [17](#)
- AstNodeBinary
 - Tang::AstNodeBinary, [20](#)

- AstNodeBlock
 - Tang::AstNodeBlock, [24](#)
- AstNodeBoolean
 - Tang::AstNodeBoolean, [27](#)
- AstNodeCast
 - Tang::AstNodeCast, [30](#)
- AstNodeFloat
 - Tang::AstNodeFloat, [33](#)
- AstNodeIdentifier
 - Tang::AstNodeIdentifier, [36](#)
- AstNodeInteger
 - Tang::AstNodeInteger, [39](#)
- AstNodeNull
 - Tang::AstNodeNull, [42](#)
- AstNodeUnary
 - Tang::AstNodeUnary, [45](#)
- BOOLEAN
 - opcode.hpp, [153](#)
- Boolean
 - Tang::AstNodeCast, [30](#)
- build/generated/location.hh, [131](#)
- CASTBOOLEAN
 - opcode.hpp, [153](#)
- CASTFLOAT
 - opcode.hpp, [153](#)
- CASTINTEGER
 - opcode.hpp, [153](#)
- CodeType
 - Tang::Program, [122](#)
- compileIdentifiers
 - Tang::AstNode, [14](#)
 - Tang::AstNodeAssign, [17](#)
 - Tang::AstNodeBinary, [21](#)
 - Tang::AstNodeBlock, [24](#)
 - Tang::AstNodeBoolean, [27](#)
 - Tang::AstNodeCast, [31](#)
 - Tang::AstNodeFloat, [33](#)
 - Tang::AstNodeIdentifier, [36](#)
 - Tang::AstNodeInteger, [39](#)
 - Tang::AstNodeNull, [42](#)
 - Tang::AstNodeUnary, [46](#)
- compileScript
 - Tang::TangBase, [127](#)
- ComputedExpressionBoolean
 - Tang::ComputedExpressionBoolean, [56](#)
- ComputedExpressionError
 - Tang::ComputedExpressionError, [65](#)
- ComputedExpressionFloat
 - Tang::ComputedExpressionFloat, [74](#)
- ComputedExpressionInteger
 - Tang::ComputedExpressionInteger, [83](#)
- DIVIDE
 - opcode.hpp, [153](#)
- Divide
 - Tang::AstNodeBinary, [20](#)
- dump
 - Tang::ComputedExpression, [52](#)
 - Tang::ComputedExpressionBoolean, [61](#)
 - Tang::ComputedExpressionError, [70](#)
 - Tang::ComputedExpressionFloat, [79](#)
 - Tang::ComputedExpressionInteger, [88](#)
 - Tang::ComputedExpressionNull, [96](#)
- dumpBytecode
 - Tang::Program, [123](#)
- DUMPPROGRAMCHECK
 - program-dumpBytecode.cpp, [174](#)
- EQ
 - opcode.hpp, [153](#)
- Equal
 - Tang::AstNodeBinary, [20](#)
- Error
 - Tang::Error, [101](#)
- error.cpp
 - operator<<, [173](#)
- execute
 - Tang::Program, [123](#)
- EXECUTEPROGRAMCHECK
 - program-execute.cpp, [176](#)
- FLOAT
 - opcode.hpp, [153](#)
- Float
 - Tang::AstNodeCast, [30](#)
- GarbageCollected
 - Tang::GarbageCollected, [104](#), [105](#)
- get
 - Tang::SingletonObjectPool< T >, [125](#)
- get_next_token
 - Tang::TangScanner, [129](#)
- getAst
 - Tang::Program, [124](#)
- getCode
 - Tang::Program, [124](#)
- getInstance
 - Tang::SingletonObjectPool< T >, [125](#)
- getResult
 - Tang::Program, [124](#)
- GreaterThan
 - Tang::AstNodeBinary, [20](#)
- GreaterThanEqual
 - Tang::AstNodeBinary, [20](#)
- GT
 - opcode.hpp, [153](#)
- GTE
 - opcode.hpp, [153](#)
- include/astNode.hpp, [133](#)
- include/astNodeAssign.hpp, [134](#)
- include/astNodeBinary.hpp, [135](#)
- include/astNodeBlock.hpp, [136](#)
- include/astNodeBoolean.hpp, [137](#)
- include/astNodeCast.hpp, [138](#)
- include/astNodeFloat.hpp, [139](#)

- include/astNodeIdentifier.hpp, 140
- include/astNodeInteger.hpp, 141
- include/astNodeNull.hpp, 142
- include/astNodeUnary.hpp, 143
- include/computedExpression.hpp, 144
- include/computedExpressionBoolean.hpp, 145
- include/computedExpressionError.hpp, 146
- include/computedExpressionFloat.hpp, 147
- include/computedExpressionInteger.hpp, 148
- include/computedExpressionNull.hpp, 149
- include/error.hpp, 150
- include/garbageCollected.hpp, 151
- include/macros.hpp, 151
- include/opcode.hpp, 152
- include/program.hpp, 153
- include/singletonObjectPool.hpp, 155
- include/tang.hpp, 156
- include/tangBase.hpp, 157
- include/tangScanner.hpp, 158
- INTEGER
 - opcode.hpp, 153
- Integer
 - Tang::AstNodeCast, 30
- is_equal
 - Tang::ComputedExpression, 52–54
 - Tang::ComputedExpressionBoolean, 61, 62
 - Tang::ComputedExpressionError, 70, 71
 - Tang::ComputedExpressionFloat, 79, 80
 - Tang::ComputedExpressionInteger, 88, 89
 - Tang::ComputedExpressionNull, 97, 98
- LessThan
 - Tang::AstNodeBinary, 20
- LessThanEqual
 - Tang::AstNodeBinary, 20
- location.hh
 - operator<<, 132, 133
- LT
 - opcode.hpp, 153
- LTE
 - opcode.hpp, 153
- macros.hpp
 - TANG_UNUSED, 152
- make
 - Tang::GarbageCollected, 105
- makeCopy
 - Tang::AstNode, 14
 - Tang::AstNodeAssign, 17
 - Tang::AstNodeBinary, 21
 - Tang::AstNodeBlock, 24
 - Tang::AstNodeBoolean, 27
 - Tang::AstNodeCast, 31
 - Tang::AstNodeFloat, 34
 - Tang::AstNodeIdentifier, 37
 - Tang::AstNodeInteger, 39
 - Tang::AstNodeNull, 42
 - Tang::AstNodeUnary, 46
 - Tang::ComputedExpression, 54
 - Tang::ComputedExpressionBoolean, 63
 - Tang::ComputedExpressionError, 72
 - Tang::ComputedExpressionFloat, 81
 - Tang::ComputedExpressionInteger, 90
 - Tang::ComputedExpressionNull, 99
- MODULO
 - opcode.hpp, 153
- Modulo
 - Tang::AstNodeBinary, 20
- MULTIPLY
 - opcode.hpp, 153
- Multiply
 - Tang::AstNodeBinary, 20
- NEGATIVE
 - opcode.hpp, 153
- Negative
 - Tang::AstNodeUnary, 45
- NEQ
 - opcode.hpp, 153
- NOT
 - opcode.hpp, 153
- Not
 - Tang::AstNodeUnary, 45
- NotEqual
 - Tang::AstNodeBinary, 20
- NULLVAL
 - opcode.hpp, 153
- Opcode
 - opcode.hpp, 153
- opcode.hpp
 - ADD, 153
 - BOOLEAN, 153
 - CASTBOOLEAN, 153
 - CASTFLOAT, 153
 - CASTINTEGER, 153
 - DIVIDE, 153
 - EQ, 153
 - FLOAT, 153
 - GT, 153
 - GTE, 153
 - INTEGER, 153
 - LT, 153
 - LTE, 153
 - MODULO, 153
 - MULTIPLY, 153
 - NEGATIVE, 153
 - NEQ, 153
 - NOT, 153
 - NULLVAL, 153
 - Opcode, 153
 - PEEK, 153
 - POKE, 153
 - POP, 153
 - SUBTRACT, 153
- Operation
 - Tang::AstNodeBinary, 20
- Operator

- Tang::AstNodeUnary, 45
- operator!
 - Tang::GarbageCollected, 106
- operator!=
 - Tang::GarbageCollected, 106
- operator<
 - Tang::GarbageCollected, 111
- operator<<
 - error.cpp, 173
 - location.hh, 132, 133
 - Tang::Error, 101
 - Tang::GarbageCollected, 117
- operator<=
 - Tang::GarbageCollected, 111
- operator>
 - Tang::GarbageCollected, 115
- operator>=
 - Tang::GarbageCollected, 115
- operator*
 - Tang::GarbageCollected, 107, 108
- operator+
 - Tang::GarbageCollected, 108
- operator-
 - Tang::GarbageCollected, 109
- operator->
 - Tang::GarbageCollected, 110
- operator/
 - Tang::GarbageCollected, 110
- operator=
 - Tang::GarbageCollected, 112
- operator==
 - Tang::GarbageCollected, 113–115
- operator%
 - Tang::GarbageCollected, 107
- PEEK
 - opcode.hpp, 153
- POKE
 - opcode.hpp, 153
- POP
 - opcode.hpp, 153
- Program
 - Tang::Program, 123
- program-dumpBytecode.cpp
 - DUMPPROGRAMCHECK, 174
- program-execute.cpp
 - EXECUTEPROGRAMCHECK, 176
 - STACKCHECK, 176
- recycle
 - Tang::SingletonObjectPool< T >, 126
- Script
 - Tang::Program, 122
- src/astNode.cpp, 159
- src/astNodeAssign.cpp, 159
- src/astNodeBinary.cpp, 160
- src/astNodeBlock.cpp, 161
- src/astNodeBoolean.cpp, 161
- src/astNodeCast.cpp, 162
- src/astNodeFloat.cpp, 163
- src/astNodeIdentifier.cpp, 164
- src/astNodeInteger.cpp, 165
- src/astNodeNull.cpp, 166
- src/astNodeUnary.cpp, 167
- src/computedExpression.cpp, 168
- src/computedExpressionBoolean.cpp, 169
- src/computedExpressionError.cpp, 170
- src/computedExpressionFloat.cpp, 170
- src/computedExpressionInteger.cpp, 171
- src/computedExpressionNull.cpp, 172
- src/error.cpp, 172
- src/program-dumpBytecode.cpp, 174
- src/program-execute.cpp, 175
- src/program.cpp, 176
- src/tangBase.cpp, 177
- STACKCHECK
 - program-execute.cpp, 176
- SUBTRACT
 - opcode.hpp, 153
- Subtract
 - Tang::AstNodeBinary, 20
- Tang::AstNode, 11
 - AstNode, 14
 - compileIdentifiers, 14
 - makeCopy, 14
- Tang::AstNodeAssign, 15
 - AstNodeAssign, 17
 - compileIdentifiers, 17
 - makeCopy, 17
- Tang::AstNodeBinary, 18
 - Add, 20
 - AstNodeBinary, 20
 - compileIdentifiers, 21
 - Divide, 20
 - Equal, 20
 - GreaterThan, 20
 - GreaterThanEqual, 20
 - LessThan, 20
 - LessThanEqual, 20
 - makeCopy, 21
 - Modulo, 20
 - Multiply, 20
 - NotEqual, 20
 - Operation, 20
 - Subtract, 20
- Tang::AstNodeBlock, 22
 - AstNodeBlock, 24
 - compileIdentifiers, 24
 - makeCopy, 24
- Tang::AstNodeBoolean, 25
 - AstNodeBoolean, 27
 - compileIdentifiers, 27
 - makeCopy, 27
- Tang::AstNodeCast, 28
 - AstNodeCast, 30
 - Boolean, 30

- compileIdentifiers, 31
- Float, 30
- Integer, 30
- makeCopy, 31
- Type, 30
- Tang::AstNodeFloat, 31
 - AstNodeFloat, 33
 - compileIdentifiers, 33
 - makeCopy, 34
- Tang::AstNodeIdentifier, 34
 - AstNodeIdentifier, 36
 - compileIdentifiers, 36
 - makeCopy, 37
- Tang::AstNodeInteger, 37
 - AstNodeInteger, 39
 - compileIdentifiers, 39
 - makeCopy, 39
- Tang::AstNodeNull, 40
 - AstNodeNull, 42
 - compileIdentifiers, 42
 - makeCopy, 42
- Tang::AstNodeUnary, 43
 - AstNodeUnary, 45
 - compileIdentifiers, 46
 - makeCopy, 46
 - Negative, 45
 - Not, 45
 - Operator, 45
- Tang::ComputedExpression, 46
 - __add, 48
 - __boolean, 48
 - __divide, 49
 - __equal, 49
 - __float, 49
 - __integer, 50
 - __lessThan, 50
 - __modulo, 50
 - __multiply, 51
 - __negative, 51
 - __not, 51
 - __subtract, 52
 - dump, 52
 - is_equal, 52–54
 - makeCopy, 54
- Tang::ComputedExpressionBoolean, 55
 - __add, 57
 - __boolean, 57
 - __divide, 57
 - __equal, 58
 - __float, 58
 - __integer, 58
 - __lessThan, 59
 - __modulo, 59
 - __multiply, 59
 - __negative, 60
 - __not, 60
 - __subtract, 60
 - ComputedExpressionBoolean, 56
- dump, 61
- is_equal, 61, 62
- makeCopy, 63
- Tang::ComputedExpressionError, 64
 - __add, 66
 - __boolean, 66
 - __divide, 66
 - __equal, 67
 - __float, 67
 - __integer, 67
 - __lessThan, 68
 - __modulo, 68
 - __multiply, 68
 - __negative, 69
 - __not, 69
 - __subtract, 69
 - ComputedExpressionError, 65
 - dump, 70
 - is_equal, 70, 71
 - makeCopy, 72
- Tang::ComputedExpressionFloat, 73
 - __add, 75
 - __boolean, 75
 - __divide, 75
 - __equal, 76
 - __float, 76
 - __integer, 76
 - __lessThan, 77
 - __modulo, 77
 - __multiply, 77
 - __negative, 78
 - __not, 78
 - __subtract, 78
 - ComputedExpressionFloat, 74
 - dump, 79
 - is_equal, 79, 80
 - makeCopy, 81
- Tang::ComputedExpressionInteger, 82
 - __add, 84
 - __boolean, 84
 - __divide, 84
 - __equal, 85
 - __float, 85
 - __integer, 85
 - __lessThan, 86
 - __modulo, 86
 - __multiply, 86
 - __negative, 87
 - __not, 87
 - __subtract, 87
 - ComputedExpressionInteger, 83
 - dump, 88
 - is_equal, 88, 89
 - makeCopy, 90
- Tang::ComputedExpressionNull, 91
 - __add, 92
 - __boolean, 93
 - __divide, 93

- __equal, 93
- __float, 94
- __integer, 94
- __lessThan, 94
- __modulo, 95
- __multiply, 95
- __negative, 95
- __not, 96
- __subtract, 96
- dump, 96
- is_equal, 97, 98
- makeCopy, 99
- Tang::Error, 99
 - Error, 101
 - operator<<, 101
- Tang::GarbageCollected, 102
 - ~GarbageCollected, 105
 - GarbageCollected, 104, 105
 - make, 105
 - operator!, 106
 - operator!=, 106
 - operator<, 111
 - operator<<, 117
 - operator<=, 111
 - operator>, 115
 - operator>=, 115
 - operator*, 107, 108
 - operator+, 108
 - operator-, 109
 - operator->, 110
 - operator/, 110
 - operator=, 112
 - operator==, 113–115
 - operator%, 107
- Tang::location, 118
- Tang::position, 119
- Tang::Program, 121
 - addBytecode, 123
 - CodeType, 122
 - dumpBytecode, 123
 - execute, 123
 - getAst, 124
 - getCode, 124
 - getResult, 124
 - Program, 123
 - Script, 122
 - Template, 122
- Tang::SingletonObjectPool< T >, 125
 - get, 125
 - getInstance, 125
 - recycle, 126
- Tang::TangBase, 126
 - compileScript, 127
 - TangBase, 127
- Tang::TangScanner, 128
 - get_next_token, 129
 - TangScanner, 129
- TANG_UNUSED
 - macros.hpp, 152
- TangBase
 - Tang::TangBase, 127
- TangScanner
 - Tang::TangScanner, 129
- Template
 - Tang::Program, 122
- test/test.cpp, 178
- test/testGarbageCollected.cpp, 179
- test/testSingletonObjectPool.cpp, 180
- Type
 - Tang::AstNodeCast, 30