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
5.2 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
5.3 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() . . . 20
5.3.4 Member Function Documentation . . . 21
5.3.4.1 compileIdentifiers() . . . 21
5.4 Tang::AstNodeBlock Class Reference . . . 21
5.4.1 Detailed Description . . . 23
5.4.2 Constructor & Destructor Documentation . . . 23
5.4.2.1 AstNodeBlock() . . . 23
5.4.3 Member Function Documentation . . . 23
5.4.3.1 compileIdentifiers() . . . 23
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
5.6 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
5.7 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() . . . 33
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
5.9 Tang::AstNodeIfElse Class Reference . . . 37
5.9.1 Detailed Description . . . 39
5.9.2 Constructor & Destructor Documentation . . . 39
5.9.2.1 AstNodeIfElse() [1/2] . . . 39
5.9.2.2 AstNodeIfElse() [2/2] . . . 39
5.9.3 Member Function Documentation . . . 40
5.9.3.1 compileIdentifiers() . . . 40
5.10 Tang::AstNodeInteger Class Reference . . . 40
5.10.1 Detailed Description . . . 42
5.10.2 Constructor & Destructor Documentation . . . 42
5.10.2.1 AstNodeInteger() . . . 42
5.10.3 Member Function Documentation . . . 42
5.10.3.1 compileIdentifiers() . . . 42
5.11 Tang::AstNodeNull Class Reference . . . 43
5.11.1 Detailed Description . . . 45
5.11.2 Constructor & Destructor Documentation . . . 45
5.11.2.1 AstNodeNull() . . . 45
5.11.3 Member Function Documentation . . . 45
5.11.3.1 compileIdentifiers() . . . 45

5.12 Tang::AstNodeUnary Class Reference . . . 45
5.12.1 Detailed Description . . . 47
5.12.2 Member Enumeration Documentation . . . 47
5.12.2.1 Operator . . . 47
5.12.3 Constructor & Destructor Documentation . . . 47
5.12.3.1 AstNodeUnary() . . . 48
5.12.4 Member Function Documentation . . . 48
5.12.4.1 compileIdentifiers() . . . 48
5.13 Tang::ComputedExpression Class Reference . . . 48
5.13.1 Detailed Description . . . 50
5.13.2 Member Function Documentation . . . 50
5.13.2.1 __add() . . . 50
5.13.2.2 __boolean() . . . 50
5.13.2.3 __divide() . . . 51
5.13.2.4 __equal() . . . 51
5.13.2.5 __float() . . . 52
5.13.2.6 __integer() . . . 52
5.13.2.7 __lessThan() . . . 52
5.13.2.8 __modulo() . . . 53
5.13.2.9 __multiply() . . . 53
5.13.2.10 __negative() . . . 53
5.13.2.11 __not() . . . 54
5.13.2.12 __subtract() . . . 54
5.13.2.13 dump() . . . 54
5.13.2.14 is_equal() [1/5] . . . 54
5.13.2.15 is_equal() [2/5] . . . 55
5.13.2.16 is_equal() [3/5] . . . 55
5.13.2.17 is_equal() [4/5] . . . 56
5.13.2.18 is_equal() [5/5] . . . 56
5.13.2.19 makeCopy() . . . 56
5.14 Tang::ComputedExpressionBoolean Class Reference . . . 57
5.14.1 Detailed Description . . . 58
5.14.2 Constructor & Destructor Documentation . . . 58
5.14.2.1 ComputedExpressionBoolean() . . . 58
5.14.3 Member Function Documentation . . . 59
5.14.3.1 __add() . . . 59
5.14.3.2 __boolean() . . . 59
5.14.3.3 __divide() . . . 59
5.14.3.4 __equal() . . . 60
5.14.3.5 __float() . . . 60
5.14.3.6 __integer() . . . 61
5.14.3.7 __lessThan() . . . 61

5.14.3.8 __modulo() . . . 61
5.14.3.9 __multiply() . . . 62
5.14.3.10 __negative() . . . 62
5.14.3.11 __not() . . . 62
5.14.3.12 __subtract() . . . 62
5.14.3.13 dump() . . . 63
5.14.3.14 is_equal() [1/5] . . . 63
5.14.3.15 is_equal() [2/5] . . . 63
5.14.3.16 is_equal() [3/5] . . . 64
5.14.3.17 is_equal() [4/5] . . . 64
5.14.3.18 is_equal() [5/5] . . . 65
5.14.3.19 makeCopy() . . . 65
5.15 Tang::ComputedExpressionError Class Reference . . . 65
5.15.1 Detailed Description . . . 67
5.15.2 Constructor & Destructor Documentation . . . 67
5.15.2.1 ComputedExpressionError() . . . 67
5.15.3 Member Function Documentation . . . 67
5.15.3.1 __add() . . . 67
5.15.3.2 __boolean() . . . 68
5.15.3.3 __divide() . . . 68
5.15.3.4 __equal() . . . 68
5.15.3.5 __float() . . . 69
5.15.3.6 __integer() . . . 69
5.15.3.7 __lessThan() . . . 69
5.15.3.8 __modulo() . . . 70
5.15.3.9 __multiply() . . . 70
5.15.3.10 __negative() . . . 71
5.15.3.11 __not() . . . 71
5.15.3.12 __subtract() . . . 71
5.15.3.13 dump() . . . 72
5.15.3.14 is_equal() [1/5] . . . 72
5.15.3.15 is_equal() [2/5] . . . 72
5.15.3.16 is_equal() [3/5] . . . 73
5.15.3.17 is_equal() [4/5] . . . 73
5.15.3.18 is_equal() [5/5] . . . 73
5.15.3.19 makeCopy() . . . 74
5.16 Tang::ComputedExpressionFloat Class Reference . . . 74
5.16.1 Detailed Description . . . 76
5.16.2 Constructor & Destructor Documentation . . . 76
5.16.2.1 ComputedExpressionFloat() . . . 76
5.16.3 Member Function Documentation . . . 76
5.16.3.1 __add() . . . 76

5.16.3.2 __boolean() . . . 77
5.16.3.3 __divide() . . . 77
5.16.3.4 __equal() . . . 77
5.16.3.5 __float() . . . 78
5.16.3.6 __integer() . . . 78
5.16.3.7 __lessThan() . . . 78
5.16.3.8 __modulo() . . . 79
5.16.3.9 __multiply() . . . 79
5.16.3.10 __negative() . . . 80
5.16.3.11 __not() . . . 80
5.16.3.12 __subtract() . . . 80
5.16.3.13 dump() . . . 81
5.16.3.14 is_equal() [1/5] . . . 81
5.16.3.15 is_equal() [2/5] . . . 81
5.16.3.16 is_equal() [3/5] . . . 82
5.16.3.17 is_equal() [4/5] . . . 82
5.16.3.18 is_equal() [5/5] . . . 82
5.16.3.19 makeCopy() . . . 83
5.17 Tang::ComputedExpressionInteger Class Reference . . . 83
5.17.1 Detailed Description . . . 85
5.17.2 Constructor & Destructor Documentation . . . 85
5.17.2.1 ComputedExpressionInteger() . . . 85
5.17.3 Member Function Documentation . . . 85
5.17.3.1 __add() . . . 85
5.17.3.2 __boolean() . . . 86
5.17.3.3 __divide() . . . 86
5.17.3.4 __equal() . . . 86
5.17.3.5 __float() . . . 87
5.17.3.6 __integer() . . . 87
5.17.3.7 __lessThan() . . . 87
5.17.3.8 __modulo() . . . 88
5.17.3.9 __multiply() . . . 88
5.17.3.10 __negative() . . . 89
5.17.3.11 __not() . . . 89
5.17.3.12 __subtract() . . . 89
5.17.3.13 dump() . . . 90
5.17.3.14 is_equal() [1/5] . . . 90
5.17.3.15 is_equal() [2/5] . . . 90
5.17.3.16 is_equal() [3/5] . . . 91
5.17.3.17 is_equal() [4/5] . . . 91
5.17.3.18 is_equal() [5/5] . . . 91
5.17.3.19 makeCopy() . . . 92

5.18 Tang::ComputedExpressionNull Class Reference . . . 92
5.18.1 Detailed Description . . . 94
5.18.2 Member Function Documentation . . . 94
5.18.2.1 __add() . . . 94
5.18.2.2 __boolean() . . . 94
5.18.2.3 __divide() . . . 95
5.18.2.4 __equal() . . . 95
5.18.2.5 __float() . . . 95
5.18.2.6 __integer() . . . 96
5.18.2.7 __lessThan() . . . 96
5.18.2.8 __modulo() . . . 96
5.18.2.9 __multiply() . . . 97
5.18.2.10 __negative() . . . 97
5.18.2.11 __not() . . . 97
5.18.2.12 __subtract() . . . 97
5.18.2.13 dump() . . . 98
5.18.2.14 is_equal() [1/5] . . . 98
5.18.2.15 is_equal() [2/5] . . . 98
5.18.2.16 is_equal() [3/5] . . . 99
5.18.2.17 is_equal() [4/5] . . . 99
5.18.2.18 is_equal() [5/5] . . . 100
5.18.2.19 makeCopy() . . . 100
5.19 Tang::Error Class Reference . . . 100
5.19.1 Detailed Description . . . 102
5.19.2 Constructor & Destructor Documentation . . . 102
5.19.2.1 Error() [1/2] . . . 102
5.19.2.2 Error() [2/2] . . . 102
5.19.3 Friends And Related Function Documentation . . . 102
5.19.3.1 operator<< . . . 103
5.20 Tang::GarbageCollected Class Reference . . . 103
5.20.1 Detailed Description . . . 105
5.20.2 Constructor & Destructor Documentation . . . 105
5.20.2.1 GarbageCollected() [1/3] . . . 105
5.20.2.2 GarbageCollected() [2/3] . . . 106
5.20.2.3 ~GarbageCollected() . . . 106
5.20.2.4 GarbageCollected() [3/3] . . . 106
5.20.3 Member Function Documentation . . . 106
5.20.3.1 make() . . . 106
5.20.3.2 operator"!() . . . 107
5.20.3.3 operator"!=() . . . 107
5.20.3.4 operator%() . . . 108
5.20.3.5 operator*() [1/2] . . . 109

5.20.3.6 operator*() [2/2] . . . 109
5.20.3.7 operator+() . . . 109
5.20.3.8 operator-() [1/2] . . . 110
5.20.3.9 operator-() [2/2] . . . 110
5.20.3.10 operator->() . . . 111
5.20.3.11 operator/() . . . 111
5.20.3.12 operator<() . . . 112
5.20.3.13 operator<=() . . . 112
5.20.3.14 operator=() [1/2] . . . 113
5.20.3.15 operator=() [2/2] . . . 113
5.20.3.16 operator==() [1/6] . . . 114
5.20.3.17 operator==() [2/6] . . . 114
5.20.3.18 operator==() [3/6] . . . 115
5.20.3.19 operator==() [4/6] . . . 115
5.20.3.20 operator==() [5/6] . . . 115
5.20.3.21 operator==() [6/6] . . . 116
5.20.3.22 operator>() . . . 116
5.20.3.23 operator>=() . . . 116
5.20.4 Friends And Related Function Documentation . . . 118
5.20.4.1 operator<< . . . 118
5.21 Tang::location Class Reference . . . 119
5.21.1 Detailed Description . . . 120
5.22 Tang::position Class Reference . . . 120
5.22.1 Detailed Description . . . 121
5.23 Tang::Program Class Reference . . . 122
5.23.1 Detailed Description . . . 123
5.23.2 Member Enumeration Documentation . . . 123
5.23.2.1 CodeType . . . 123
5.23.3 Constructor & Destructor Documentation . . . 123
5.23.3.1 Program() . . . 124
5.23.4 Member Function Documentation . . . 124
5.23.4.1 addBytecode() . . . 124
5.23.4.2 dumpBytecode() . . . 124
5.23.4.3 execute() . . . 125
5.23.4.4 getAst() . . . 125
5.23.4.5 getBytecode() . . . 125
5.23.4.6 getCode() . . . 125
5.23.4.7 getResult() . . . 126
5.23.4.8 setJumpTarget() . . . 126
5.24 Tang::SingletonObjectPool< T > Class Template Reference . . . 126
5.24.1 Detailed Description . . . 127
5.24.2 Member Function Documentation . . . 127

5.24.2.1 get() . . . 127
5.24.2.2 getInstance() . . . 127
5.24.2.3 recycle() . . . 127
5.25 Tang::TangBase Class Reference . . . 128
5.25.1 Detailed Description . . . 128
5.25.2 Constructor & Destructor Documentation . . . 128
5.25.2.1 TangBase() . . . 128
5.25.3 Member Function Documentation . . . 128
5.25.3.1 compileScript() . . . 128
5.26 Tang::TangScanner Class Reference . . . 129
5.26.1 Detailed Description . . . 130
5.26.2 Constructor & Destructor Documentation . . . 130
5.26.2.1 TangScanner() . . . 130
5.26.3 Member Function Documentation . . . 130
5.26.3.1 get_next_token() . . . 130

6 File Documentation **133**

6.1 build/generated/location.hh File Reference . . . 133
6.1.1 Detailed Description . . . 134
6.1.2 Function Documentation . . . 134
6.1.2.1 operator<<() [1/2] . . . 134
6.1.2.2 operator<<() [2/2] . . . 135
6.2 include/astNode.hpp File Reference . . . 135
6.2.1 Detailed Description . . . 136
6.3 include/astNodeAssign.hpp File Reference . . . 136
6.3.1 Detailed Description . . . 137
6.4 include/astNodeBinary.hpp File Reference . . . 137
6.4.1 Detailed Description . . . 138
6.5 include/astNodeBlock.hpp File Reference . . . 138
6.5.1 Detailed Description . . . 139
6.6 include/astNodeBoolean.hpp File Reference . . . 139
6.6.1 Detailed Description . . . 140
6.7 include/astNodeCast.hpp File Reference . . . 140
6.7.1 Detailed Description . . . 141
6.8 include/astNodeFloat.hpp File Reference . . . 141
6.8.1 Detailed Description . . . 142
6.9 include/astNodeIdentifier.hpp File Reference . . . 142
6.9.1 Detailed Description . . . 143
6.10 include/astNodeIfElse.hpp File Reference . . . 143
6.10.1 Detailed Description . . . 144
6.11 include/astNodeInteger.hpp File Reference . . . 144
6.11.1 Detailed Description . . . 145

6.12 include/astNodeNull.hpp File Reference . . . 145
6.12.1 Detailed Description . . . 146
6.13 include/astNodeUnary.hpp File Reference . . . 146
6.13.1 Detailed Description . . . 147
6.14 include/computedExpression.hpp File Reference . . . 147
6.14.1 Detailed Description . . . 148
6.15 include/computedExpressionBoolean.hpp File Reference . . . 148
6.15.1 Detailed Description . . . 149
6.16 include/computedExpressionError.hpp File Reference . . . 149
6.16.1 Detailed Description . . . 150
6.17 include/computedExpressionFloat.hpp File Reference . . . 150
6.17.1 Detailed Description . . . 150
6.18 include/computedExpressionInteger.hpp File Reference . . . 151
6.18.1 Detailed Description . . . 151
6.19 include/computedExpressionNull.hpp File Reference . . . 152
6.19.1 Detailed Description . . . 152
6.20 include/error.hpp File Reference . . . 153
6.20.1 Detailed Description . . . 153
6.21 include/garbageCollected.hpp File Reference . . . 154
6.21.1 Detailed Description . . . 154
6.22 include/macros.hpp File Reference . . . 154
6.22.1 Detailed Description . . . 155
6.22.2 Macro Definition Documentation . . . 155
6.22.2.1 TANG_UNUSED . . . 155
6.23 include/opcode.hpp File Reference . . . 155
6.23.1 Detailed Description . . . 156
6.23.2 Enumeration Type Documentation . . . 156
6.23.2.1 Opcode . . . 156
6.24 include/program.hpp File Reference . . . 156
6.24.1 Detailed Description . . . 157
6.25 include/singletonObjectPool.hpp File Reference . . . 158
6.25.1 Detailed Description . . . 158
6.26 include/tang.hpp File Reference . . . 159
6.26.1 Detailed Description . . . 159
6.27 include/tangBase.hpp File Reference . . . 160
6.27.1 Detailed Description . . . 161
6.28 include/tangScanner.hpp File Reference . . . 161
6.28.1 Detailed Description . . . 162
6.29 src/astNode.cpp File Reference . . . 162
6.29.1 Detailed Description . . . 162
6.30 src/astNodeAssign.cpp File Reference . . . 162
6.30.1 Detailed Description . . . 163

6.31 src/astNodeBinary.cpp File Reference . . . 163
6.31.1 Detailed Description . . . 164
6.32 src/astNodeBlock.cpp File Reference . . . 164
6.32.1 Detailed Description . . . 164
6.33 src/astNodeBoolean.cpp File Reference . . . 164
6.33.1 Detailed Description . . . 165
6.34 src/astNodeCast.cpp File Reference . . . 165
6.34.1 Detailed Description . . . 166
6.35 src/astNodeFloat.cpp File Reference . . . 166
6.35.1 Detailed Description . . . 167
6.36 src/astNodeIdentifier.cpp File Reference . . . 167
6.36.1 Detailed Description . . . 168
6.37 src/astNodeIfElse.cpp File Reference . . . 168
6.37.1 Detailed Description . . . 169
6.38 src/astNodeInteger.cpp File Reference . . . 169
6.38.1 Detailed Description . . . 170
6.39 src/astNodeNull.cpp File Reference . . . 170
6.39.1 Detailed Description . . . 170
6.40 src/astNodeUnary.cpp File Reference . . . 170
6.40.1 Detailed Description . . . 171
6.41 src/computedExpression.cpp File Reference . . . 171
6.41.1 Detailed Description . . . 172
6.42 src/computedExpressionBoolean.cpp File Reference . . . 172
6.42.1 Detailed Description . . . 173
6.43 src/computedExpressionError.cpp File Reference . . . 173
6.43.1 Detailed Description . . . 173
6.44 src/computedExpressionFloat.cpp File Reference . . . 173
6.44.1 Detailed Description . . . 174
6.45 src/computedExpressionInteger.cpp File Reference . . . 174
6.45.1 Detailed Description . . . 174
6.46 src/computedExpressionNull.cpp File Reference . . . 175
6.46.1 Detailed Description . . . 175
6.47 src/error.cpp File Reference . . . 175
6.47.1 Detailed Description . . . 176
6.47.2 Function Documentation . . . 176
6.47.2.1 operator<<() . . . 176
6.48 src/program-dumpBytecode.cpp File Reference . . . 177
6.48.1 Detailed Description . . . 177
6.48.2 Macro Definition Documentation . . . 177
6.48.2.1 DUMPPROGRAMCHECK . . . 178
6.49 src/program-execute.cpp File Reference . . . 178
6.49.1 Detailed Description . . . 179

6.49.2 Macro Definition Documentation 179
6.49.2.1 EXECUTEPROGRAMCHECK 179
6.49.2.2 STACKCHECK 179
6.50 src/program.cpp File Reference 179
6.50.1 Detailed Description 180
6.51 src/tangBase.cpp File Reference 180
6.51.1 Detailed Description 181
6.52 test/test.cpp File Reference 181
6.52.1 Detailed Description 182
6.53 test/testGarbageCollected.cpp File Reference 182
6.53.1 Detailed Description 182
6.54 test/testSingletonObjectPool.cpp File Reference 183
6.54.1 Detailed Description 183

Index **185**

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 . . . 21
- Tang::AstNodeBoolean . . . 25
- Tang::AstNodeCast . . . 28
- Tang::AstNodeFloat . . . 31
- Tang::AstNodeIdentifier . . . 34
- Tang::AstNodeIfElse . . . 37
- Tang::AstNodeInteger . . . 40
- Tang::AstNodeNull . . . 43
- Tang::AstNodeUnary . . . 45

Tang::ComputedExpression . . . 48
- Tang::ComputedExpressionBoolean . . . 57
- Tang::ComputedExpressionError . . . 65
- Tang::ComputedExpressionFloat . . . 74
- Tang::ComputedExpressionInteger . . . 83
- Tang::ComputedExpressionNull . . . 92

Tang::Error . . . 100
Tang::GarbageCollected . . . 103
Tang::location . . . 119
Tang::position . . . 120
Tang::Program . . . 122
Tang::SingletonObjectPool< T > . . . 126
Tang::TangBase . . . 128
TangTangFlexLexer
- Tang::TangScanner . . . 129

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 . . . 21

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::AstNodeIfElse
: An AstNode that represents an if..else statement . . . 37

Tang::AstNodeInteger
: An AstNode that represents an integer literal . . . 40

Tang::AstNodeNull
: An AstNode that represents a NULL value . . . 43

Tang::AstNodeUnary
: An AstNode that represents a unary negation . . . 45

Tang::ComputedExpression
: Represents the result of a computation that has been executed . . . 48

Tang::ComputedExpressionBoolean
: Represents an Boolean that is the result of a computation . . . 57

Tang::ComputedExpressionError
: Represents a Runtime Error . . . 65

Tang::ComputedExpressionFloat
: Represents a Float that is the result of a computation . . . 74

Tang::ComputedExpressionInteger
: Represents an Integer that is the result of a computation . . . 83

Tang::ComputedExpressionNull
: Represents an Null that is the result of a computation . . . 92

Tang::Error
Used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error . . . 100
Tang::GarbageCollected
A container that acts as a resource-counting garbage collector for the specified type . . . 103
Tang::location
Two points in a source file . . . 119
Tang::position
A point in a source file . . . 120
Tang::Program
Represents a compiled script or template that may be executed . . . 122
Tang::SingletonObjectPool< T >
A thread-safe, singleton object pool of the designated type . . . 126
Tang::TangBase
The base class for the Tang programming language . . . 128
Tang::TangScanner
The Flex lexer class for the main Tang language . . . 129

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 Tang ::location class . . . 133
include/astNode.hpp
Declare the Tang::AstNode base class . . . 135
include/astNodeAssign.hpp
Declare the Tang::AstNodeAssign class . . . 136
include/astNodeBinary.hpp
Declare the Tang::AstNodeBinary class . . . 137
include/astNodeBlock.hpp
Declare the Tang::AstNodeBlock class . . . 138
include/astNodeBoolean.hpp
Declare the Tang::AstNodeBoolean class . . . 139
include/astNodeCast.hpp
Declare the Tang::AstNodeCast class . . . 140
include/astNodeFloat.hpp
Declare the Tang::AstNodeFloat class . . . 141
include/astNodeIdentifier.hpp
Declare the Tang::AstNodeIdentifier class . . . 142
include/astNodeIfElse.hpp
Declare the Tang::AstNodeIfElse class . . . 143
include/astNodeInteger.hpp
Declare the Tang::AstNodeInteger class . . . 144
include/astNodeNull.hpp
Declare the Tang::AstNodeNull class . . . 145
include/astNodeUnary.hpp
Declare the Tang::AstNodeUnary class . . . 146
include/computedExpression.hpp
Declare the Tang::ComputedExpression base class . . . 147
include/computedExpressionBoolean.hpp
Declare the Tang::ComputedExpressionBoolean class . . . 148
include/computedExpressionError.hpp
Declare the Tang::ComputedExpressionError class . . . 149
include/computedExpressionFloat.hpp
Declare the Tang::ComputedExpressionFloat class . . . 150
include/computedExpressionInteger.hpp
Declare the Tang::ComputedExpressionInteger class . . . 151

include/computedExpressionNull.hpp
Declare the Tang::ComputedExpressionNull class . . . 152
include/error.hpp
Declare the Tang::Error class used to describe syntax and runtime errors . . . 153
include/garbageCollected.hpp
Declare the Tang::GarbageCollected class . . . 154
include/macros.hpp
Contains generic macros . . . 154
include/opcode.hpp
Declare the Opcodes used in the Bytecode representation of a program . . . 155
include/program.hpp
Declare the Tang::Program class used to compile and execute source code . . . 156
include/singletonObjectPool.hpp
Declare the Tang::SingletonObjectPool class . . . 158
include/tang.hpp
Header file supplied for use by 3rd party code so that they can easily include all necessary headers . . . 159
include/tangBase.hpp
Declare the Tang::TangBase class used to interact with Tang . . . 160
include/tangScanner.hpp
Declare the Tang::TangScanner used to tokenize a Tang script . . . 161
src/astNode.cpp
Define the Tang::AstNode class . . . 162
src/astNodeAssign.cpp
Define the Tang::AstNodeAssign class . . . 162
src/astNodeBinary.cpp
Define the Tang::AstNodeBinary class . . . 163
src/astNodeBlock.cpp
Define the Tang::AstNodeBlock class . . . 164
src/astNodeBoolean.cpp
Define the Tang::AstNodeBoolean class . . . 164
src/astNodeCast.cpp
Define the Tang::AstNodeCast class . . . 165
src/astNodeFloat.cpp
Define the Tang::AstNodeFloat class . . . 166
src/astNodeIdentifier.cpp
Define the Tang::AstNodeIdentifier class . . . 167
src/astNodeIfElse.cpp
Define the Tang::AstNodeIfElse class . . . 168
src/astNodeInteger.cpp
Define the Tang::AstNodeInteger class . . . 169
src/astNodeNull.cpp
Define the Tang::AstNodeNull class . . . 170
src/astNodeUnary.cpp
Define the Tang::AstNodeUnary class . . . 170
src/computedExpression.cpp
Define the Tang::ComputedExpression class . . . 171
src/computedExpressionBoolean.cpp
Define the Tang::ComputedExpressionBoolean class . . . 172
src/computedExpressionError.cpp
Define the Tang::ComputedExpressionError class . . . 173
src/computedExpressionFloat.cpp
Define the Tang::ComputedExpressionFloat class . . . 173
src/computedExpressionInteger.cpp
Define the Tang::ComputedExpressionInteger class . . . 174
src/computedExpressionNull.cpp
Define the Tang::ComputedExpressionNull class . . . 175

src/error.cpp
Define the Tang::Error class . . . 175
src/program-dumpBytecode.cpp
Define the Tang::Program::dumpBytecode method . . . 177
src/program-execute.cpp
Define the Tang::Program::execute method . . . 178
src/program.cpp
Define the Tang::Program class . . . 179
src/tangBase.cpp
Define the Tang::TangBase class . . . 180
test/test.cpp
Test the general language behaviors . . . 181
test/testGarbageCollected.cpp
Test the generic behavior of the Tang::GarbageCollected class . . . 182
test/testSingletonObjectPool.cpp
Test the generic behavior of the Tang::SingletonObjectPool class . . . 183

Chapter 5

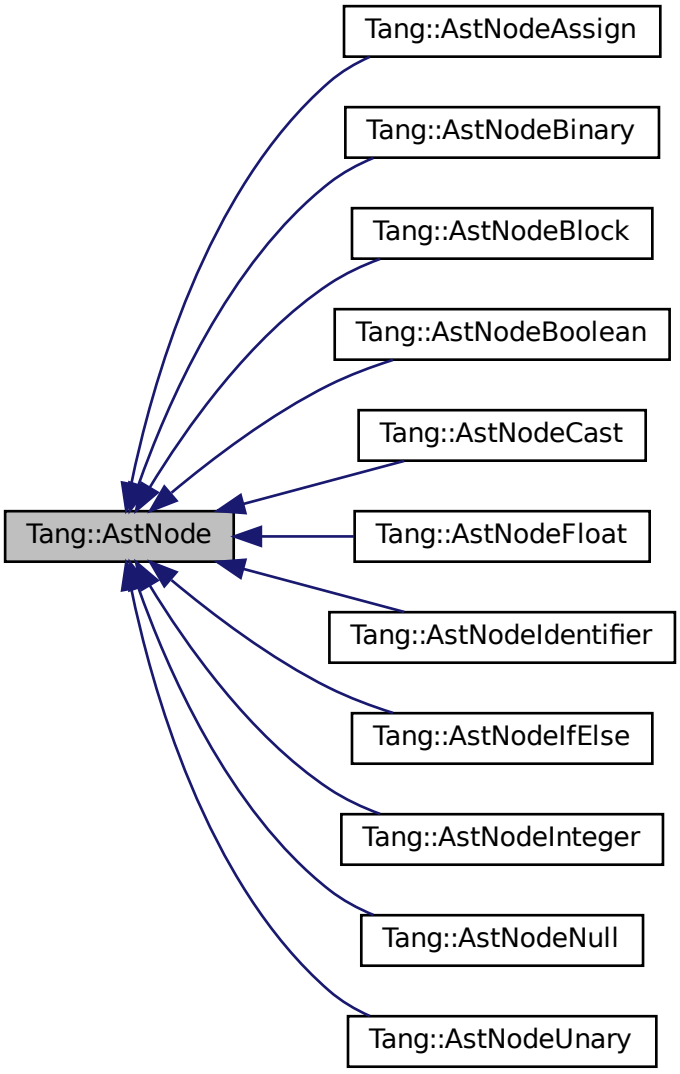
Class Documentation

5.1 Tang::AstNode Class Reference

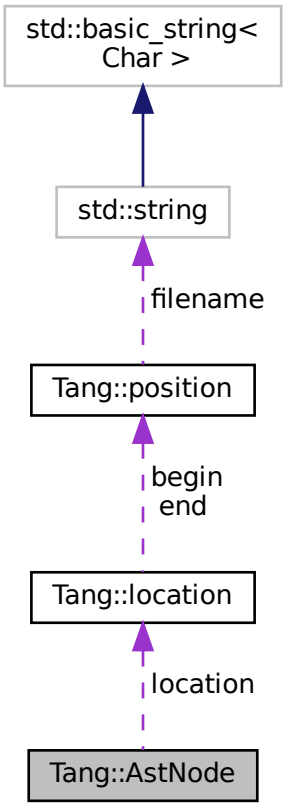
Base class for representing nodes of an Abstract Syntax Tree (AST).

`#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 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

location	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

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

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

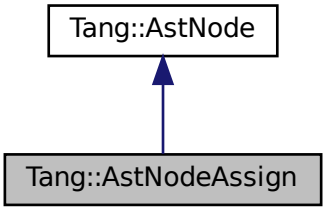
- include/astNode.hpp
- src/astNode.cpp

5.2 Tang::AstNodeAssign Class Reference

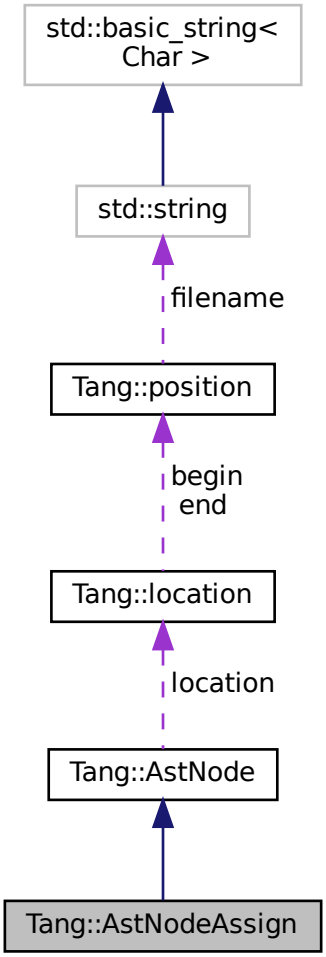
An AstNode that represents a binary expression.

`#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 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

lhs	The left hand side expression.
rhs	The right hand side expression.
location	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

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

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

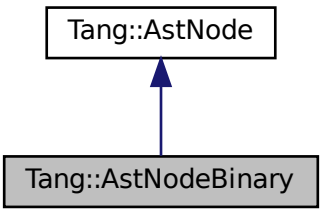
- include/astNodeAssign.hpp
- src/astNodeAssign.cpp

5.3 Tang::AstNodeBinary Class Reference

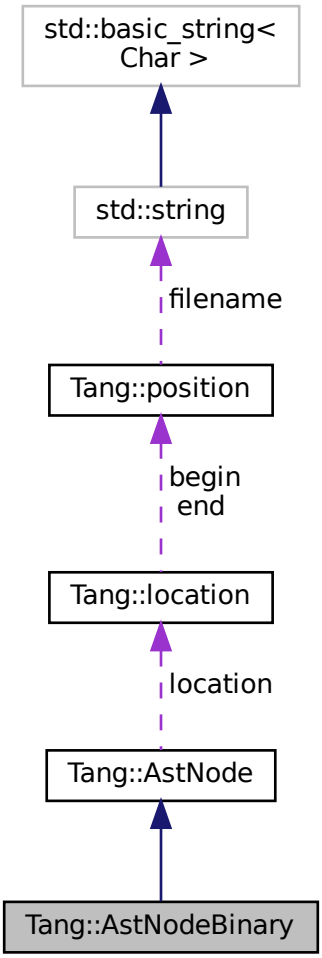
An AstNode that represents a binary expression.

`#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 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

op	The Tang::AstNodeBinary::Operation to perform.
lhs	The left hand side expression.
rhs	The right hand side expression.
location	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

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

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

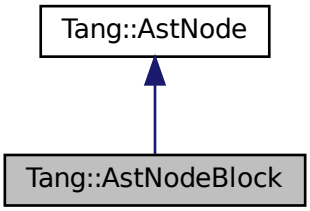
- include/astNodeBinary.hpp
- src/astNodeBinary.cpp

5.4 Tang::AstNodeBlock Class Reference

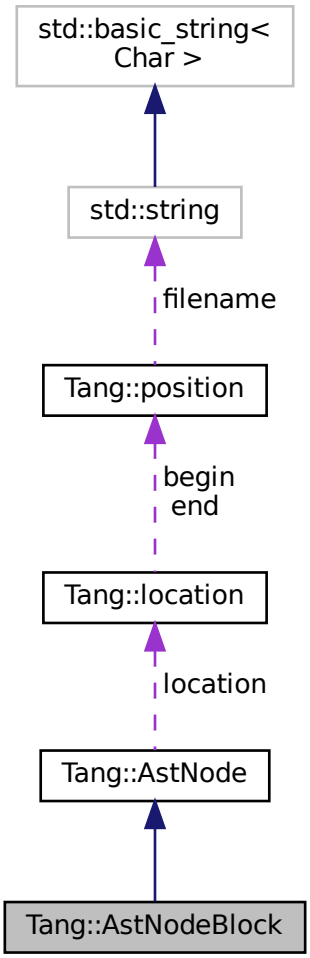
An AstNode that represents a code block.

```
#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 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

statements	The statements of the code block.
location	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

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

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

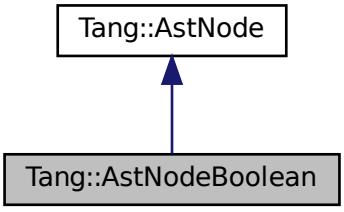
- include/astNodeBlock.hpp
- src/astNodeBlock.cpp

5.5 Tang::AstNodeBoolean Class Reference

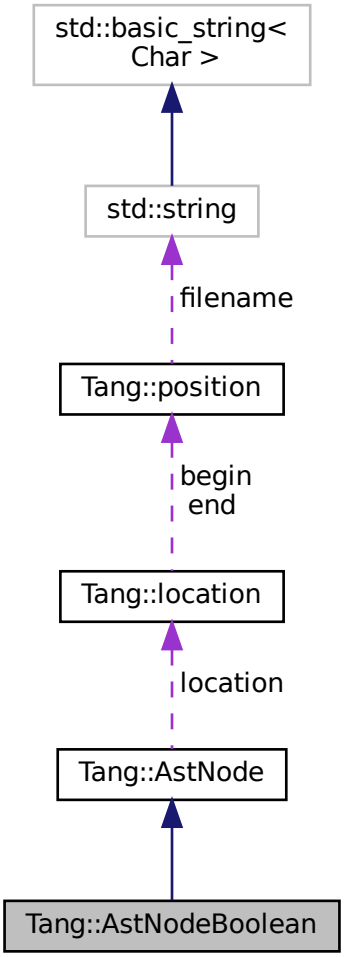
An AstNode that represents a boolean literal.

`#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 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

val	The boolean to represent.
location	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

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

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

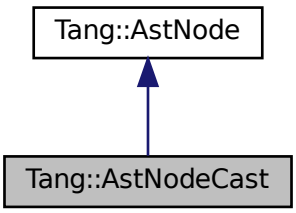
- include/astNodeBoolean.hpp
- src/astNodeBoolean.cpp

5.6 Tang::AstNodeCast Class Reference

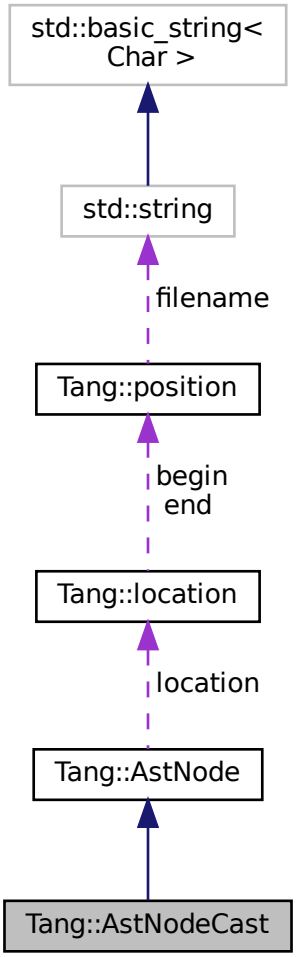
An AstNode that represents a typecast of an expression.

`#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 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

targetType	The target type that the expression will be cast to.
expression	The expression to be typecast.
location	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

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

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

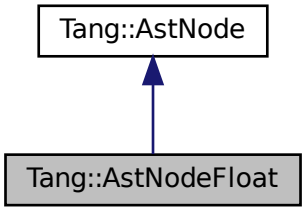
- include/astNodeCast.hpp
- src/astNodeCast.cpp

5.7 Tang::AstNodeFloat Class Reference

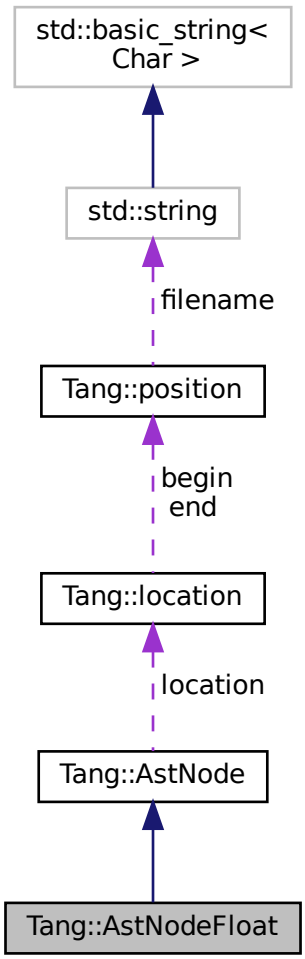
An AstNode that represents an float literal.

```
#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 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

number	The number to represent.
location	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

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

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

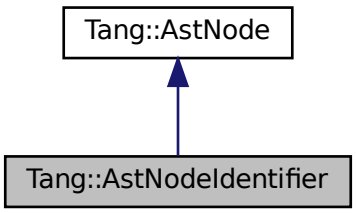
- include/astNodeFloat.hpp
- src/astNodeFloat.cpp

5.8 Tang::AstNodeIdentifier Class Reference

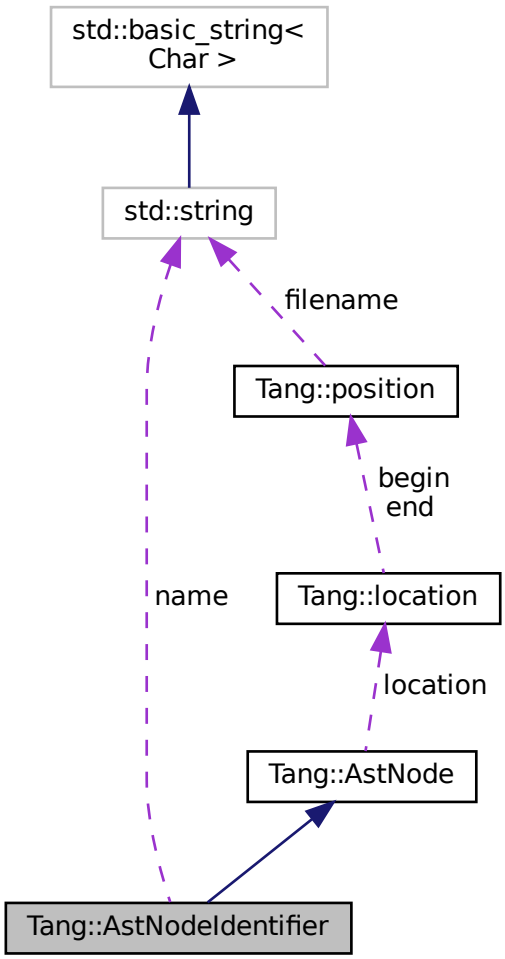
An AstNode that represents an identifier.

`#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 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

name	The name of the identifier
location	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

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

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

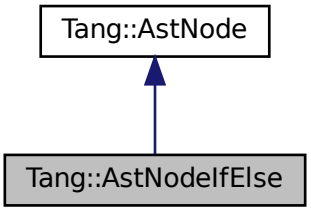
- include/astNodeIdentifier.hpp
- src/astNodeIdentifier.cpp

5.9 Tang::AstNodeIfElse Class Reference

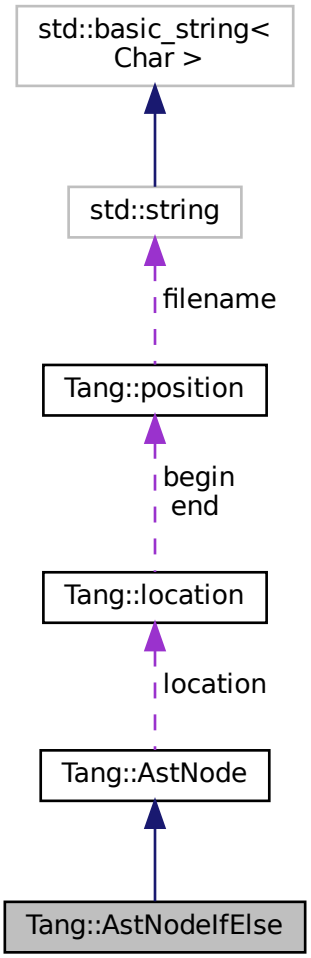
An AstNode that represents an if..else statement.

`#include <astNodeIfElse.hpp>`

Inheritance diagram for Tang::AstNodeIfElse:



Collaboration diagram for Tang::AstNodeIfElse:



Public Member Functions

- AstNodeIfElse (shared_ptr< AstNode > condition, shared_ptr< AstNode > thenBlock, shared_ptr< AstNode > elseBlock, Tang::location location)

  The constructor.
- AstNodeIfElse (shared_ptr< AstNode > condition, shared_ptr< AstNode > thenBlock, 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 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.

5.9.1 Detailed Description

An AstNode that represents an if..else statement.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 AstNodeIfElse() [1/2]

```
AstNodeIfElse::AstNodeIfElse (
            shared_ptr< AstNode > condition,
            shared_ptr< AstNode > thenBlock,
            shared_ptr< AstNode > elseBlock,
            Tang::location location )
```

The constructor.

Parameters

condition	The expression which determines whether the thenBlock or elseBlock is executed.
thenBlock	The statement executed when the condition is true.
elseBlock	The statement executed when the condition is false.
location	The location associated with the expression.

5.9.2.2 AstNodeIfElse() [2/2]

```
AstNodeIfElse::AstNodeIfElse (
            shared_ptr< AstNode > condition,
            shared_ptr< AstNode > thenBlock,
            Tang::location location )
```

The constructor.

Parameters

condition	The expression which determines whether the thenBlock or elseBlock is executed.
thenBlock	The statement executed when the condition is true.
location	The location associated with the expression.

5.9.3 Member Function Documentation

5.9.3.1 compileIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

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

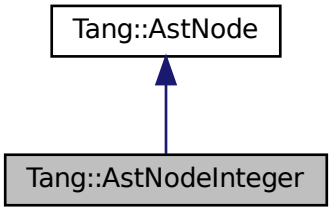
- include/astNodeIfElse.hpp
- src/astNodeIfElse.cpp

5.10 Tang::AstNodeInteger Class Reference

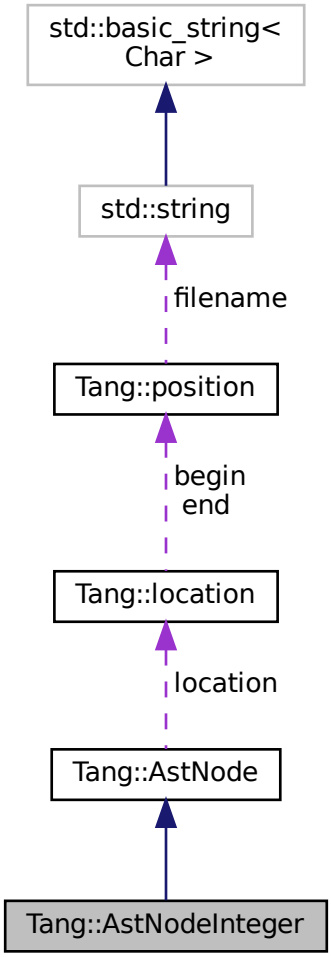
An AstNode that represents an integer literal.

```
#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 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 an integer literal.

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

5.10.2 Constructor & Destructor Documentation

5.10.2.1 AstNodeInteger()

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

The constructor.

Parameters

number	The number to represent.
location	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

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

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

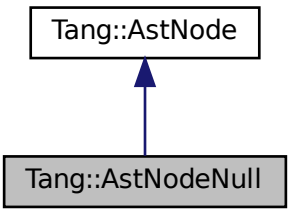
- include/astNodeInteger.hpp
- src/astNodeInteger.cpp

5.11 Tang::AstNodeNull Class Reference

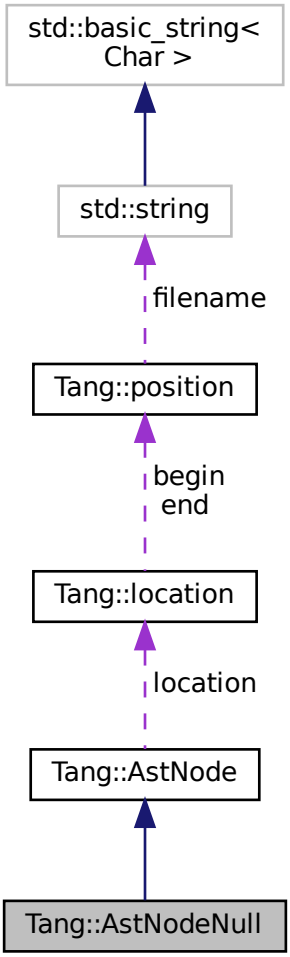
An AstNode that represents a NULL value.

`#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 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 NULL value.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 AstNodeNull()

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

The constructor.

Parameters

location	The location associated with the expression.

5.11.3 Member Function Documentation

5.11.3.1 compileIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

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

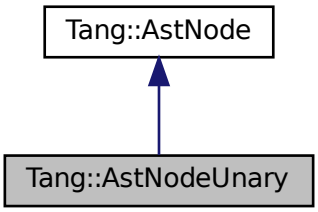
- include/astNodeNull.hpp
- src/astNodeNull.cpp

5.12 Tang::AstNodeUnary Class Reference

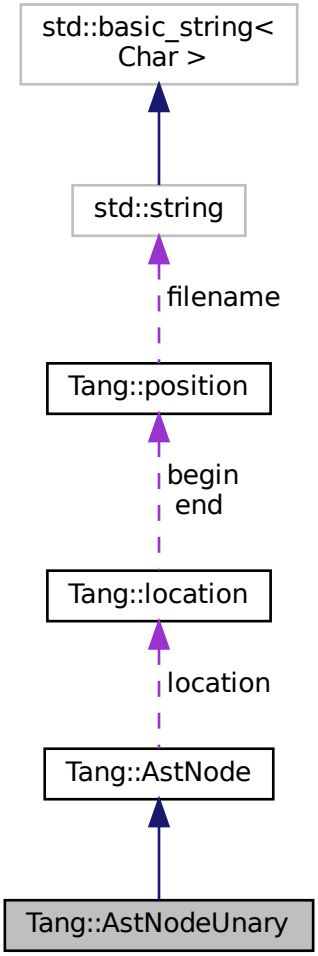
An AstNode that represents a unary negation.

`#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 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.

5.12.1 Detailed Description

An AstNode that represents a unary negation.

5.12.2 Member Enumeration Documentation

5.12.2.1 Operator

`enum Tang::AstNodeUnary::Operator`

The type of operation.

Enumerator

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

5.12.3 Constructor & Destructor Documentation

5.12.3.1 AstNodeUnary()

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

The constructor.

Parameters

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

5.12.4 Member Function Documentation

5.12.4.1 compileIdentifiers()

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

Compile a list of all variables in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

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

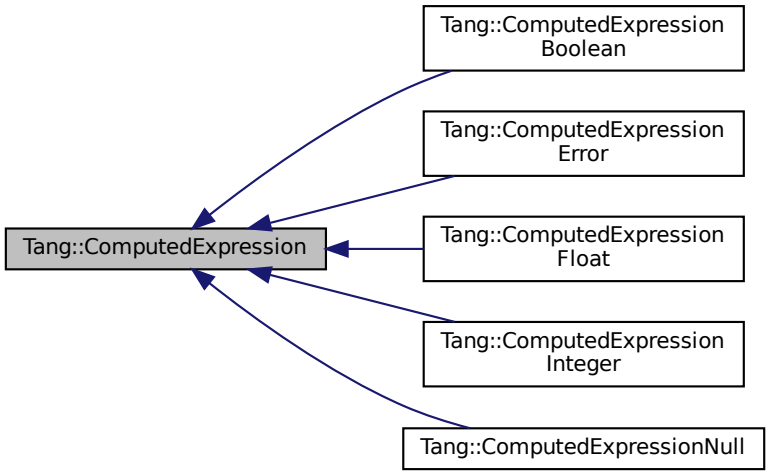
- include/astNodeUnary.hpp
- src/astNodeUnary.cpp

5.13 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 GarbageCollected 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.

5.13.1 Detailed Description

Represents the result of a computation that has been executed.

5.13.2 Member Function Documentation

5.13.2.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

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

5.13.2.2 __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.

5.13.2.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

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

5.13.2.4 __equal()

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

Perform an equalit test.

Parameters

rhs	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.

5.13.2.5 __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.

5.13.2.6 __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.

5.13.2.7 __lessThan()

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

Compute the "less than" comparison.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

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

5.13.2.8 __modulo()

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

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

Parameters

rhs	The GarbageCollected value to modulo this by.

Returns

The result of the operation.

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

5.13.2.9 __multiply()

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

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

Parameters

rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

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

5.13.2.10 __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.

5.13.2.11 __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.

5.13.2.12 __subtract()

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

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

Parameters

rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

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

5.13.2.13 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.

5.13.2.14 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

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionBoolean.

5.13.2.15 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

val	The value to compare against.

Returns

True if equal, false if not.

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

5.13.2.16 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

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionError.

5.13.2.17 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

val	The value to compare against.

Returns

True if equal, false if not.

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

5.13.2.18 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

val	The value to compare against.

Returns

True if equal, false if not.

5.13.2.19 makeCopy()

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

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

Returns

A Tang::GarbageCollected value for 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:

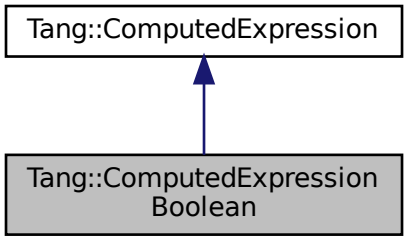
- include/computedExpression.hpp
- src/computedExpression.cpp

5.14 Tang::ComputedExpressionBoolean Class Reference

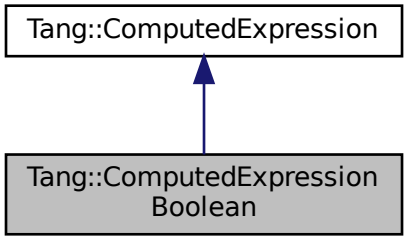
Represents an Boolean that is the result of a computation.

`#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.
- GarbageCollected 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.

5.14.1 Detailed Description

Represents an Boolean that is the result of a computation.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 ComputedExpressionBoolean()

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

Construct an Boolean result.

Parameters

val	The boolean value.

5.14.3 Member Function Documentation

5.14.3.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

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

5.14.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.14.3.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

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

5.14.3.4 __equal()

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

Perform an equalit test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.14.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.14.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.14.3.7 __lessThan()

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

Compute the "less than" comparison.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

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

5.14.3.8 __modulo()

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

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

Parameters

rhs	The GarbageCollected value to modulo this by.

Returns

The result of the operation.

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

5.14.3.9 __multiply()

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

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

Parameters

rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

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

5.14.3.10 __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.

5.14.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.14.3.12 __subtract()

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

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

Parameters

rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

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

5.14.3.13 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.

5.14.3.14 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

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

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

val	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]

```
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

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionError.

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

val	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

val	The value to compare against.

Returns

True if equal, false if not.

5.14.3.19 makeCopy()

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

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

Returns

A Tang::GarbageCollected value for the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

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

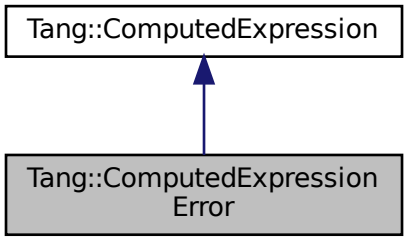
- include/computedExpressionBoolean.hpp
- src/computedExpressionBoolean.cpp

5.15 Tang::ComputedExpressionError Class Reference

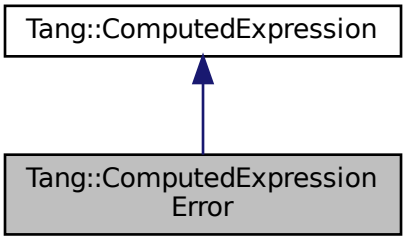
Represents a Runtime Error.

`#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.
- GarbageCollected 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.

5.15.1 Detailed Description

Represents a Runtime Error.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 ComputedExpressionError()

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

Construct a Runtime Error.

Parameters

error	The Tang::Error object.

5.15.3 Member Function Documentation

5.15.3.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.15.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.15.3.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.15.3.4 __equal()

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

Perform an equalit test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.15.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.15.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.15.3.7 __lessThan()

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

Compute the "less than" comparison.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.15.3.8 __modulo()

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

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

Parameters

rhs	The GarbageCollected value to modulo this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.15.3.9 __multiply()

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

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

Parameters

rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.15.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.15.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.15.3.12 __subtract()

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

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

Parameters

rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.15.3.13 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.

5.15.3.14 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

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionBoolean.

5.15.3.15 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

val	The value to compare against.

Returns

True if equal, false if not.

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

5.15.3.16 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

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.15.3.17 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

val	The value to compare against.

Returns

True if equal, false if not.

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

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

val	The value to compare against.

Returns

True if equal, false if not.

5.15.3.19 makeCopy()

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

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

Returns

A Tang::GarbageCollected value for the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

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

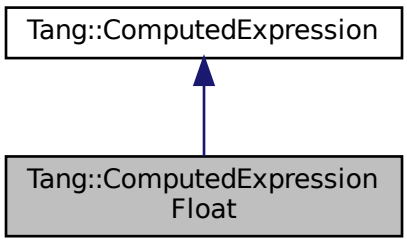
- include/computedExpressionError.hpp
- src/computedExpressionError.cpp

5.16 Tang::ComputedExpressionFloat Class Reference

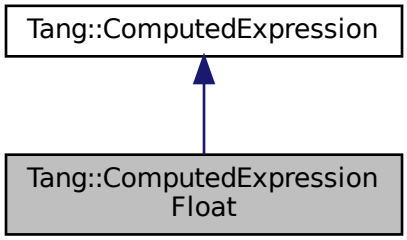
Represents a Float that is the result of a computation.

```
#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.
- GarbageCollected 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 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.
- virtual GarbageCollected __modulo (const GarbageCollected &rhs) const

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

Friends

- class **ComputedExpressionInteger**

5.16.1 Detailed Description

Represents a Float that is the result of a computation.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 ComputedExpressionFloat()

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

Construct a Float result.

Parameters

val	The float value.

5.16.3 Member Function Documentation

5.16.3.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.16.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.16.3.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.16.3.4 __equal()

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

Perform an equalit test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.16.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.16.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.16.3.7 __lessThan()

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

Compute the "less than" comparison.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.16.3.8 __modulo()

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

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

Parameters

rhs	The GarbageCollected value to modulo this by.

Returns

The result of the operation.

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

5.16.3.9 __multiply()

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

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

Parameters

rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.16.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.16.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.16.3.12 __subtract()

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

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

Parameters

rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.16.3.13 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.

5.16.3.14 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

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

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

Parameters

val	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

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

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

Parameters

val	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

val	The value to compare against.

Returns

True if equal, false if not.

5.16.3.19 makeCopy()

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

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

Returns

A Tang::GarbageCollected value for the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

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

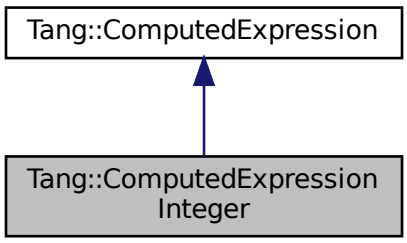
- include/computedExpressionFloat.hpp
- src/computedExpressionFloat.cpp

5.17 Tang::ComputedExpressionInteger Class Reference

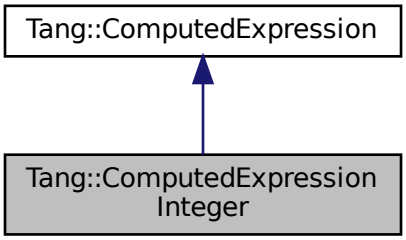
Represents an Integer that is the result of a computation.

```
#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.
- GarbageCollected 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**

5.17.1 Detailed Description

Represents an Integer that is the result of a computation.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 ComputedExpressionInteger()

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

Construct an Integer result.

Parameters

val	The integer value.

5.17.3 Member Function Documentation

5.17.3.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.17.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.17.3.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.17.3.4 __equal()

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

Perform an equalit test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.17.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.17.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.17.3.7 __lessThan()

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

Compute the "less than" comparison.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.17.3.8 __modulo()

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

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

Parameters

rhs	The GarbageCollected value to modulo this by.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.17.3.9 __multiply()

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

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

Parameters

rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.17.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.17.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.17.3.12 __subtract()

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

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

Parameters

rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.17.3.13 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.

5.17.3.14 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

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionBoolean.

5.17.3.15 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

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.17.3.16 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

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionError.

5.17.3.17 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

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented from Tang::ComputedExpression.

5.17.3.18 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

val	The value to compare against.

Returns

True if equal, false if not.

5.17.3.19 makeCopy()

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

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

Returns

A Tang::GarbageCollected value for the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

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

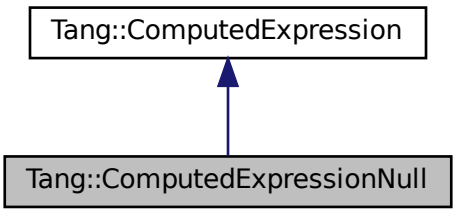
- include/computedExpressionInteger.hpp
- src/computedExpressionInteger.cpp

5.18 Tang::ComputedExpressionNull Class Reference

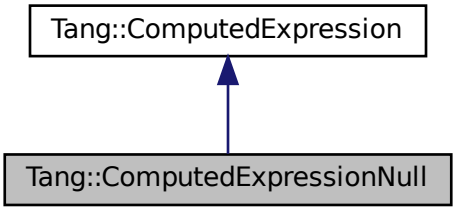
Represents an Null that is the result of a computation.

```
#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.
- GarbageCollected 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.

5.18.1 Detailed Description

Represents an Null that is the result of a computation.

5.18.2 Member Function Documentation

5.18.2.1 __add()

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

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

Parameters

rhs	The GarbageCollected value to add to this.

Returns

The result of the operation.

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

5.18.2.2 __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.

5.18.2.3 __divide()

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

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

Parameters

rhs	The GarbageCollected value to divide this by.

Returns

The result of the operation.

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

5.18.2.4 __equal()

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

Perform an equalit test.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.18.2.5 __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.

5.18.2.6 __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.

5.18.2.7 __lessThan()

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

Compute the "less than" comparison.

Parameters

rhs	The GarbageCollected value to compare against.

Returns

The result of the the operation.

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

5.18.2.8 __modulo()

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

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

Parameters

rhs	The GarbageCollected value to modulo this by.

Returns

The result of the operation.

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

5.18.2.9 __multiply()

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

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

Parameters

rhs	The GarbageCollected value to multiply to this.

Returns

The result of the operation.

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

5.18.2.10 __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.

5.18.2.11 __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.

5.18.2.12 __subtract()

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

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

Parameters

rhs	The GarbageCollected value to subtract from this.

Returns

The result of the operation.

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

5.18.2.13 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.

5.18.2.14 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

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionBoolean.

5.18.2.15 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

val	The value to compare against.

Returns

True if equal, false if not.

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

5.18.2.16 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

val	The value to compare against.

Returns

True if equal, false if not.

Reimplemented in Tang::ComputedExpressionError.

5.18.2.17 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

val	The value to compare against.

Returns

True if equal, false if not.

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

5.18.2.18 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

val	The value to compare against.

Returns

True if equal, false if not.

5.18.2.19 makeCopy()

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

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

Returns

A Tang::GarbageCollected value for the new ComputedExpression.

Reimplemented from Tang::ComputedExpression.

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

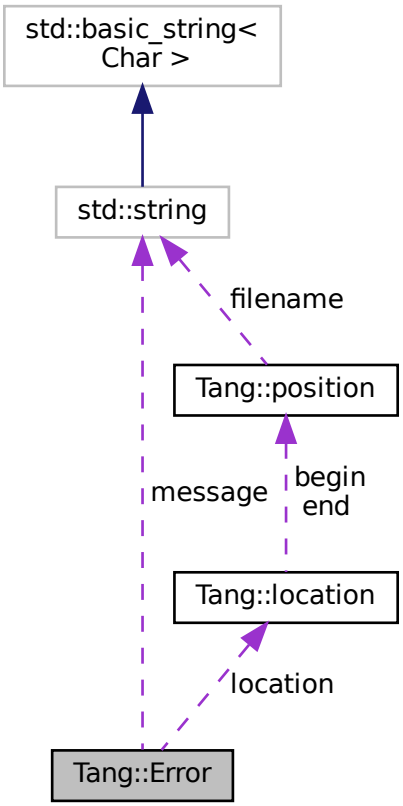
- include/computedExpressionNull.hpp
- src/computedExpressionNull.cpp

5.19 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.

5.19.1 Detailed Description

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

5.19.2 Constructor & Destructor Documentation

5.19.2.1 Error() [1/2]

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

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

Parameters

message	The error message as a string.

5.19.2.2 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

message	The error message as a string.
location	The location of the error.

5.19.3 Friends And Related Function Documentation

5.19.3.1 operator<<

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

Add friendly output.

Parameters

out	The output stream.
error	The Error object.

Returns

The output stream.

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

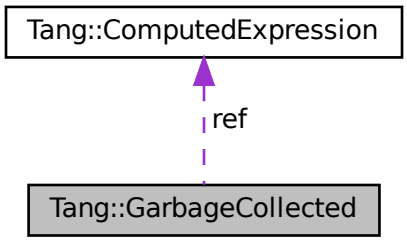
- include/error.hpp
- src/error.cpp

5.20 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.

5.20.1 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.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 GarbageCollected() [1/3]

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

Copy Constructor.

Parameters

The	other GarbageCollected object to copy.

5.20.2.2 GarbageCollected() [2/3]

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

Move Constructor.

Parameters

The	other GarbageCollected object to move.

5.20.2.3 ~GarbageCollected()

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

Destructor.

Clean up the tracked object, if appropriate.

5.20.2.4 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

variable	The arguments to pass to the constructor of the specified type.

5.20.3 Member Function Documentation

5.20.3.1 make()

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

Creates a garbage-collected object of the specified type.

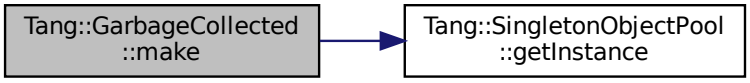
Parameters

variable	The arguments to pass to the constructor of the specified type.

Returns

A GarbageCollected object.

Here is the call graph for this function:



5.20.3.2 operator"!()

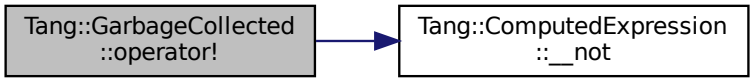
```
GarbageCollected GarbageCollected::operator!  ( ) const
```

Perform a logical not on the GarbageCollected value.

Returns

The result of the operation.

Here is the call graph for this function:



5.20.3.3 operator"!=()

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

Perform a != between two GarbageCollected values.

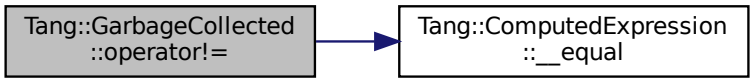
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.20.3.4 operator%()

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

Perform a modulo between two GarbageCollected values.

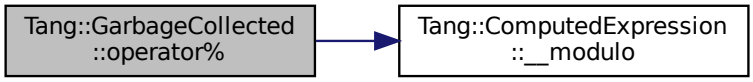
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.20.3.5 operator*() [1/2]

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

Access the tracked object.

Returns

A reference to the tracked object.

5.20.3.6 operator*() [2/2]

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

Perform a multiplication between two GarbageCollected values.

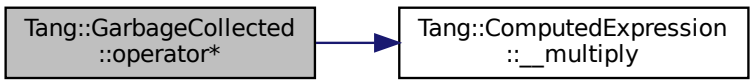
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.20.3.7 operator+()

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

Perform an addition between two GarbageCollected values.

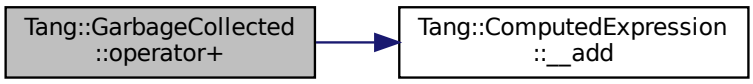
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.20.3.8 operator-() [1/2]

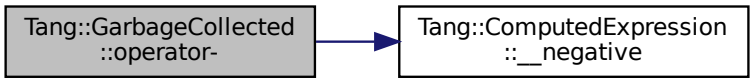
```
GarbageCollected GarbageCollected::operator- ( ) const
```

Perform a negation on the GarbageCollected value.

Returns

The result of the operation.

Here is the call graph for this function:



5.20.3.9 operator-() [2/2]

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

Perform a subtraction between two GarbageCollected values.

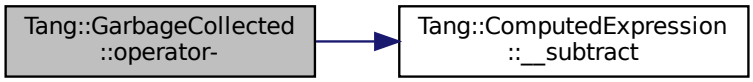
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.20.3.10 operator->()

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

Access the tracked object as a pointer.

Returns

A pointer to the tracked object.

5.20.3.11 operator/()

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

Perform a division between two GarbageCollected values.

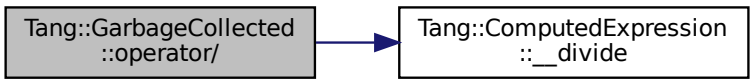
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.20.3.12 operator<()

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

Perform a < between two GarbageCollected values.

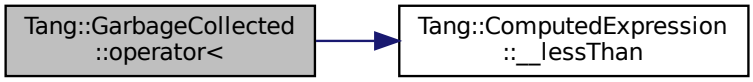
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.20.3.13 operator<=()

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

Perform a <= between two GarbageCollected values.

Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

5.20.3.14 operator=() [1/2]

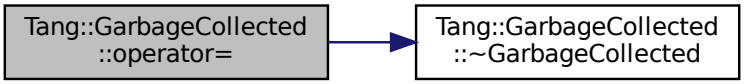
```
GarbageCollected& Tang::GarbageCollected::operator= (
            const GarbageCollected & other )  [inline]
```

Copy Assignment.

Parameters

The	other GarbageCollected object.

Here is the call graph for this function:



5.20.3.15 operator=() [2/2]

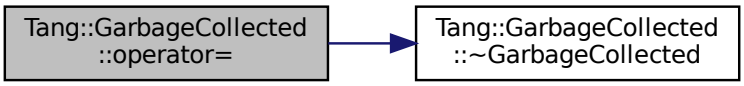
```
GarbageCollected& Tang::GarbageCollected::operator= (
            GarbageCollected && other )  [inline]
```

Move Assignment.

Parameters

The	other GarbageCollected object.

Here is the call graph for this function:



5.20.3.16 operator==() [1/6]

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

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.20.3.17 operator==() [2/6]

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

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.20.3.18 operator==() [3/6]

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

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.20.3.19 operator==() [4/6]

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

Perform a == between two GarbageCollected values.

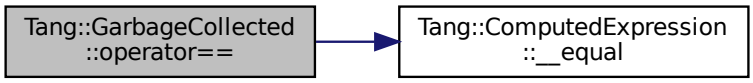
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.20.3.20 operator==() [5/6]

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

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.20.3.21 operator==() [6/6]

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

Compare the GarbageCollected tracked object with a supplied value.

Parameters

val	The value to compare the tracked object against.

Returns

True if they are equal, false otherwise.

5.20.3.22 operator>()

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

Perform a > between two GarbageCollected values.

Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

5.20.3.23 operator>=()

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

Perform a >= between two GarbageCollected values.

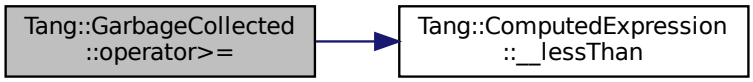
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.20.4 Friends And Related Function Documentation

5.20.4.1 operator<<

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

Add friendly output.

Parameters

out	The output stream.
gc	The GarbageCollected value.

Returns

The output stream.

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

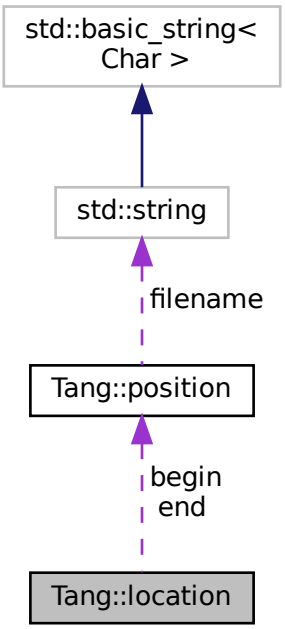
- include/garbageCollected.hpp
- src/garbageCollected.cpp

5.21 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.

5.21.1 Detailed Description

Two points in a source file.

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

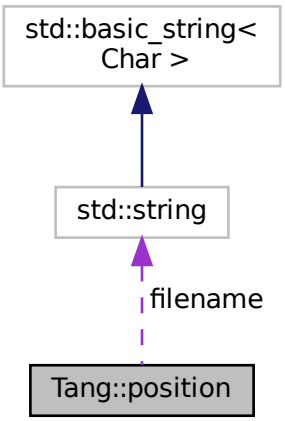
- build/generated/location.hh

5.22 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.

5.22.1 Detailed Description

A point in a source file.

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

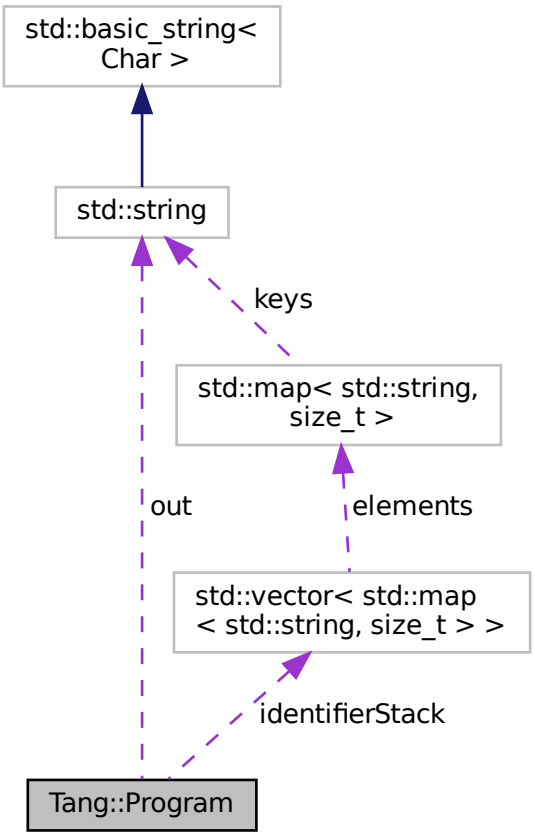
- build/generated/location.hh

5.23 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.
- 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.
- size_t addBytecode (uint64_t)

  Add a uint64_t to the Bytecode.
- const Bytecode & getBytecode ()

  Get the Bytecode vector.
- Program & execute ()

  Execute the program's Bytecode, and return the current Program object.
- bool setJumpTarget (size_t opcodeAddress, uint64_t jumpTarget)

  Set the target address of a Jump opcode.

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.

5.23.1 Detailed Description

Represents a compiled script or template that may be executed.

5.23.2 Member Enumeration Documentation

5.23.2.1 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.

5.23.3 Constructor & Destructor Documentation

5.23.3.1 Program()

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

Create a compiled program using the provided code.

Parameters

code	The code to be compiled.
codeType	Whether the code is a `Script` or `Template`.

5.23.4 Member Function Documentation

5.23.4.1 addBytecode()

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

Add a uint64_t to the Bytecode.

Parameters

op	The value to add to the Bytecode.

Returns

The size of the bytecode structure.

5.23.4.2 dumpBytecode()

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

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

Returns

A string containing the Opcode representation.

5.23.4.3 execute()

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

Execute the program's Bytecode, and return the current Program object.

Returns

The current Program object.

5.23.4.4 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.

5.23.4.5 getBytecode()

```
const Bytecode & Program::getBytecode ( )
```

Get the Bytecode vector.

Returns

The Bytecode vector.

5.23.4.6 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.

5.23.4.7 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.

5.23.4.8 setJumpTarget()

```
bool Program::setJumpTarget (
            size_t opcodeAddress,
            uint64_t jumpTarget )
```

Set the target address of a Jump opcode.

Parameters

opcodeAddress	The location of the jump statement.
jumpTarget	The address to jump to.

Returns

Whether or not the jumpTarget was set.

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

5.24 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.

5.24.1 Detailed Description

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

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

5.24.2 Member Function Documentation

5.24.2.1 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.

5.24.2.2 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.

5.24.2.3 recycle()

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

Recycle a memory location for an object T.

Parameters

obj	The memory location to recycle.

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

- include/singletonObjectPool.hpp

5.25 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.

5.25.1 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).

5.25.2 Constructor & Destructor Documentation

5.25.2.1 TangBase()

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

The constructor.

Isn't it glorious.

5.25.3 Member Function Documentation

5.25.3.1 compileScript()

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

Compile the provided source code as a script and return a Program.

Parameters

script	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:

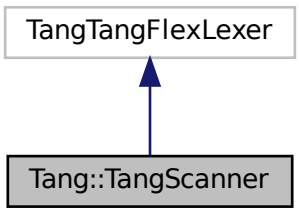
- include/tangBase.hpp
- src/tangBase.cpp

5.26 Tang::TangScanner Class Reference

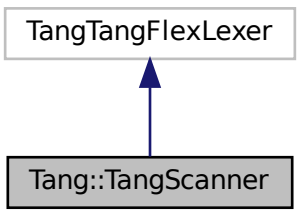
The Flex lexer class for the main Tang language.

`#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.

5.26.1 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 "TangTang↩ FlexLexer". We are subclassing it so that we can override the return type of get_next_token(), for compatibility with Bison 3 tokens.

5.26.2 Constructor & Destructor Documentation

5.26.2.1 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

arg_yyin	The input stream to be tokenized
arg_yyout	The output stream (not currently used)

5.26.3 Member Function Documentation

5.26.3.1 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

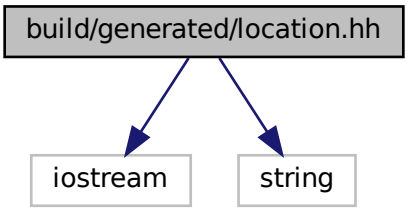
File Documentation

6.1 build/generated/location.hh File Reference

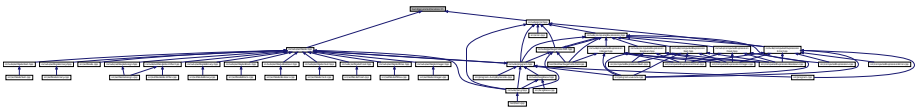
Define the Tang ::location class.

```
#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

ostr	the destination output stream
loc	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

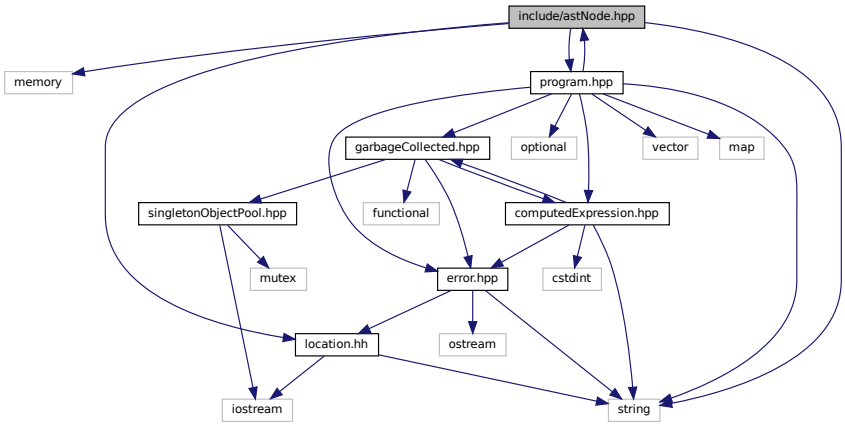
ostr	the destination output stream
pos	a reference to the position to redirect

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

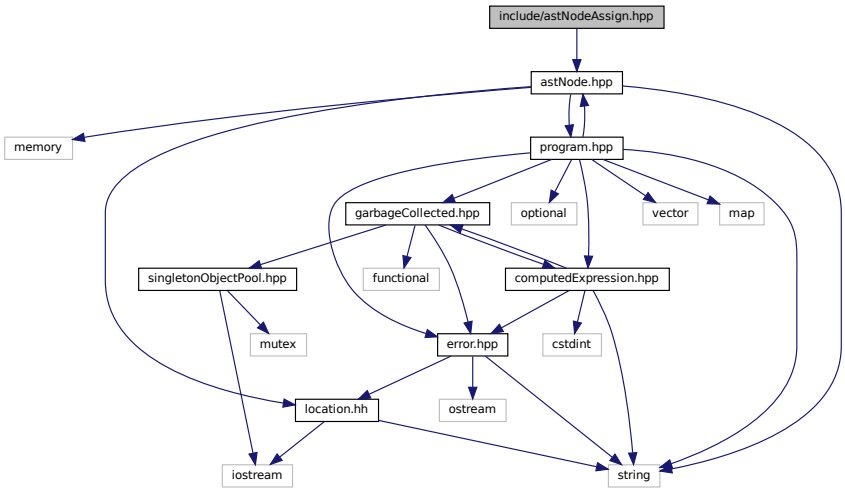
Declare the Tang::AstNode base class.

6.3 include/astNodeAssign.hpp File Reference

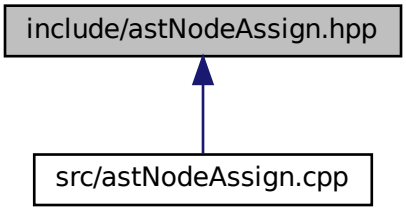
Declare the Tang::AstNodeAssign class.

```
#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

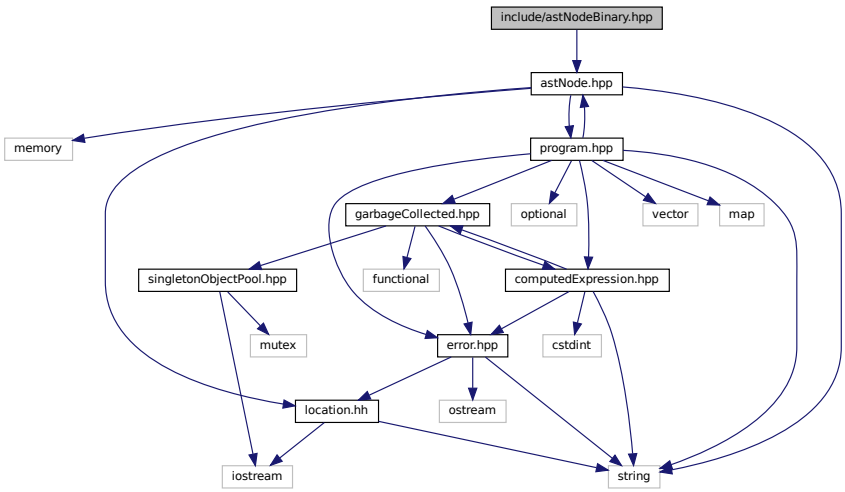
Declare the Tang::AstNodeAssign class.

6.4 include/astNodeBinary.hpp File Reference

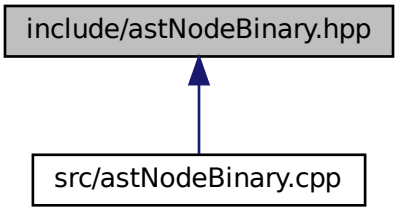
Declare the Tang::AstNodeBinary class.

```
#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

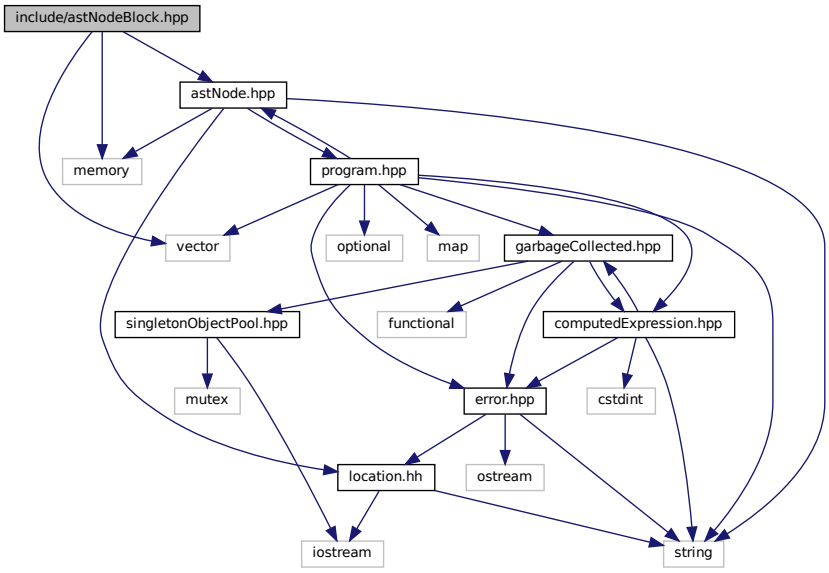
Declare the Tang::AstNodeBinary class.

6.5 include/astNodeBlock.hpp File Reference

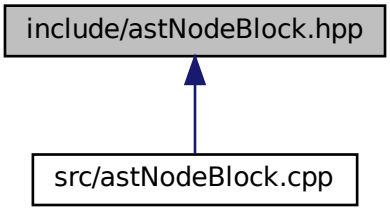
Declare the Tang::AstNodeBlock class.

```
#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

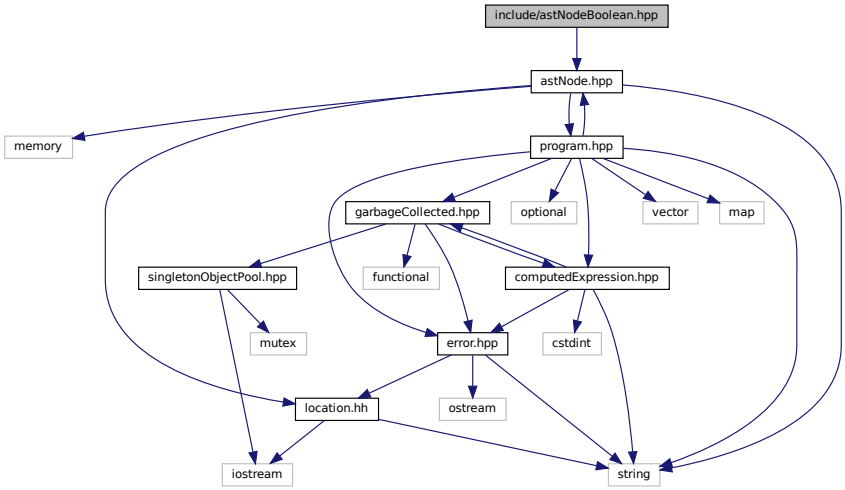
Declare the Tang::AstNodeBlock class.

6.6 include/astNodeBoolean.hpp File Reference

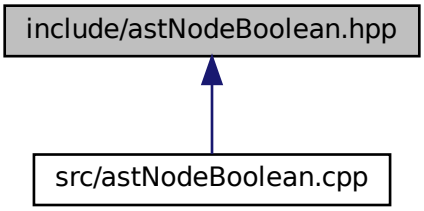
Declare the Tang::AstNodeBoolean class.

```
#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

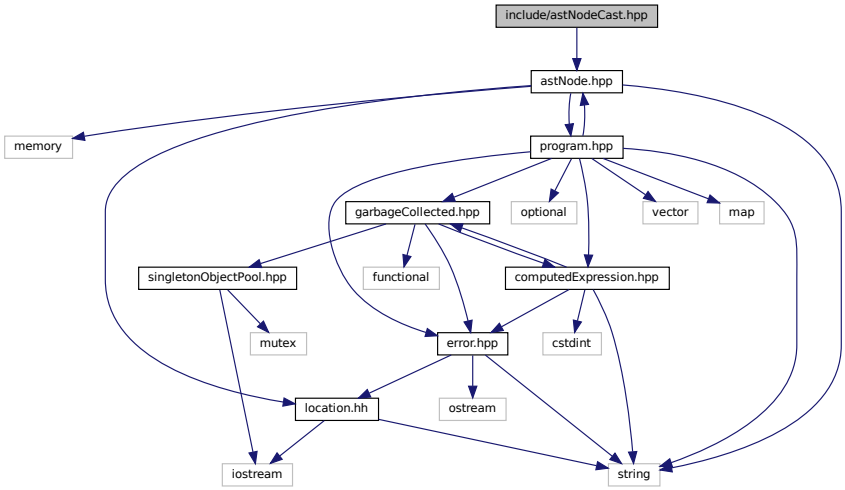
Declare the Tang::AstNodeBoolean class.

6.7 include/astNodeCast.hpp File Reference

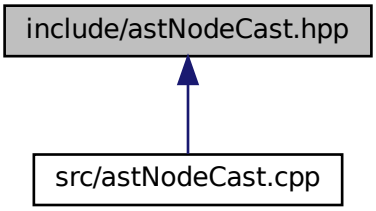
Declare the Tang::AstNodeCast class.

```
#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.

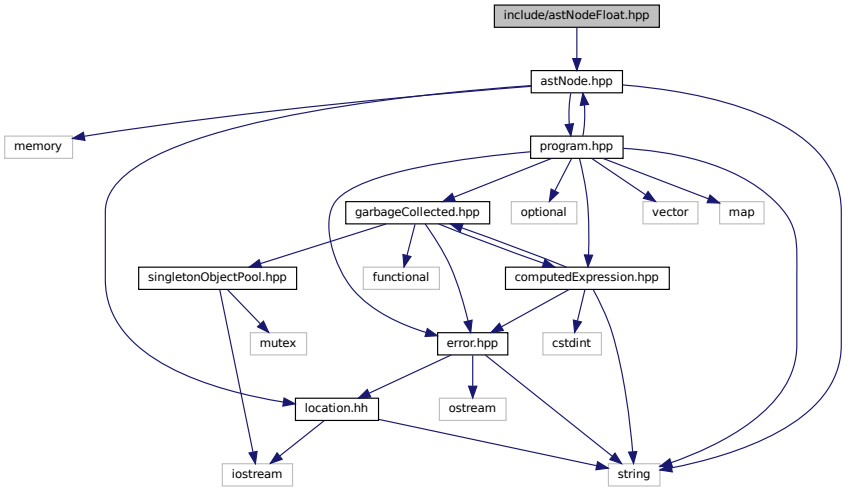
6.7.1 Detailed Description

Declare the Tang::AstNodeCast class.

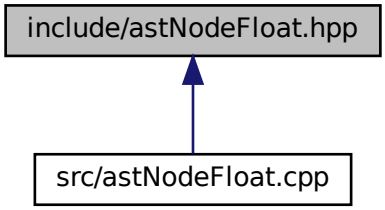
6.8 include/astNodeFloat.hpp File Reference

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

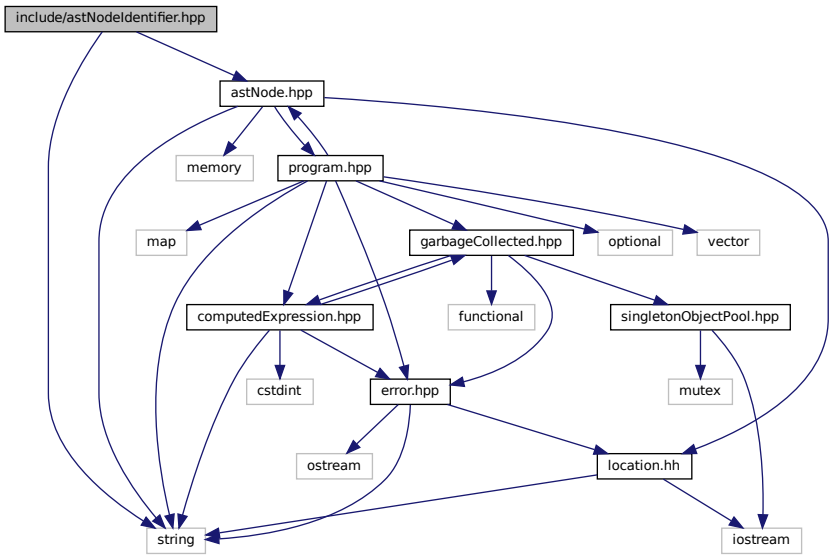
Declare the Tang::AstNodeFloat class.

6.9 include/astNodeIdentifier.hpp File Reference

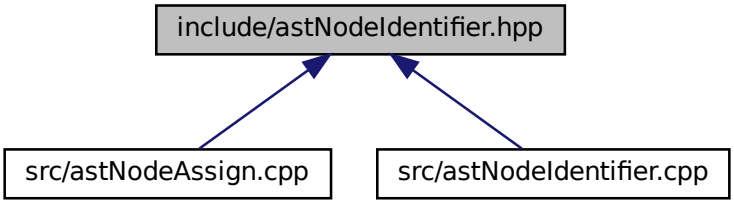
Declare the Tang::AstNodeIdentifier class.

```
#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

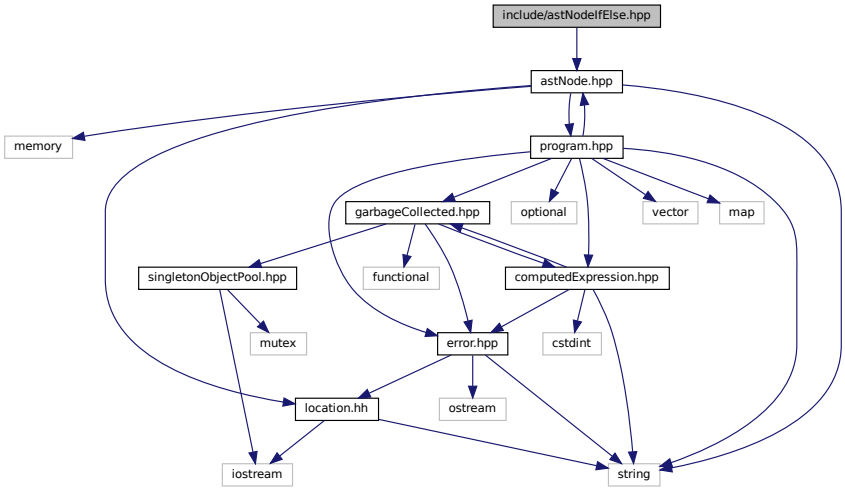
Declare the Tang::AstNodeIdentifier class.

6.10 include/astNodeIfElse.hpp File Reference

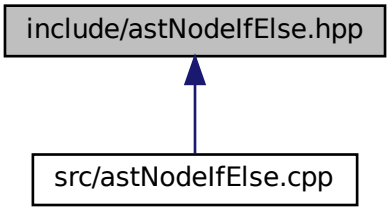
Declare the Tang::AstNodeIfElse class.

```
#include "astNode.hpp"
```

Include dependency graph for astNodeIfElse.hpp:



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



Classes

- class Tang::AstNodeIfElse

  An AstNode that represents an if..else statement.

6.10.1 Detailed Description

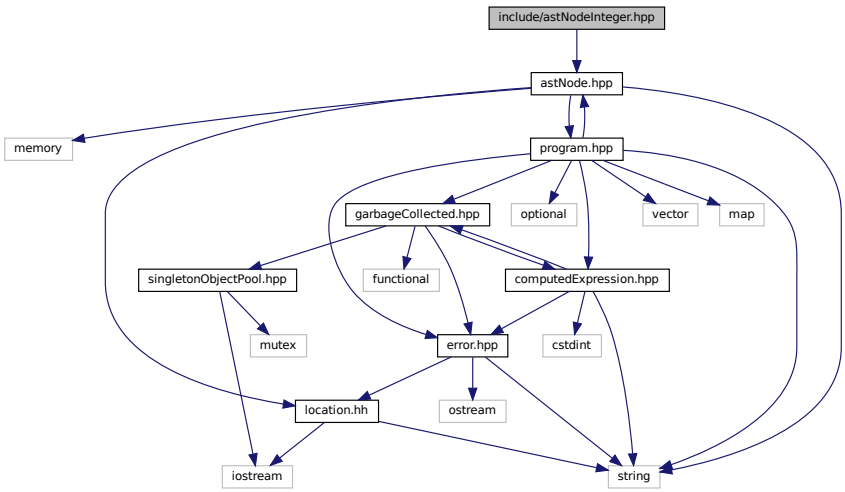
Declare the Tang::AstNodeIfElse class.

6.11 include/astNodeInteger.hpp File Reference

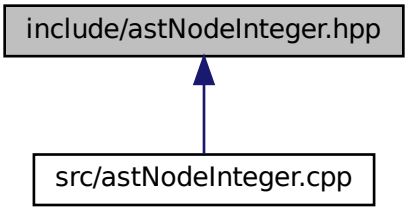
Declare the Tang::AstNodeInteger class.

```
#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.

6.11.1 Detailed Description

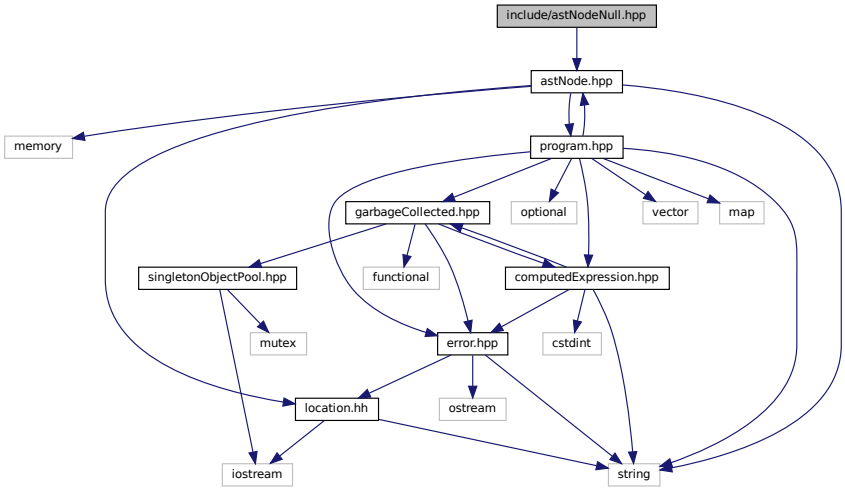
Declare the Tang::AstNodeInteger class.

6.12 include/astNodeNull.hpp File Reference

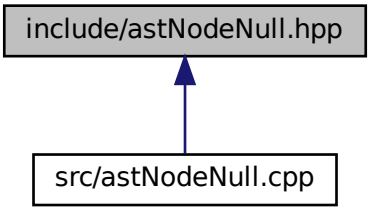
Declare the Tang::AstNodeNull class.

```
#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.

6.12.1 Detailed Description

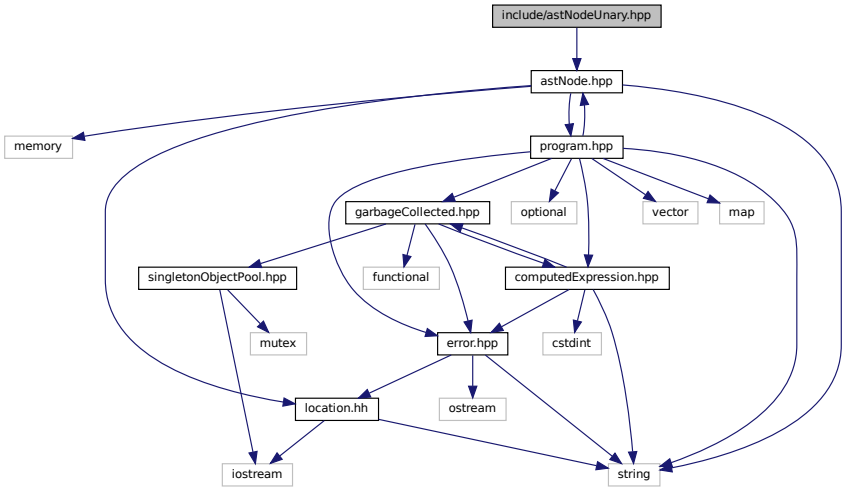
Declare the Tang::AstNodeNull class.

6.13 include/astNodeUnary.hpp File Reference

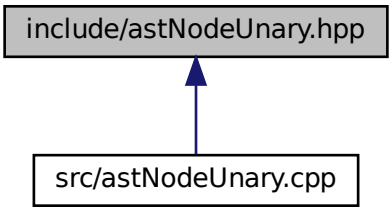
Declare the Tang::AstNodeUnary class.

```
#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.

6.13.1 Detailed Description

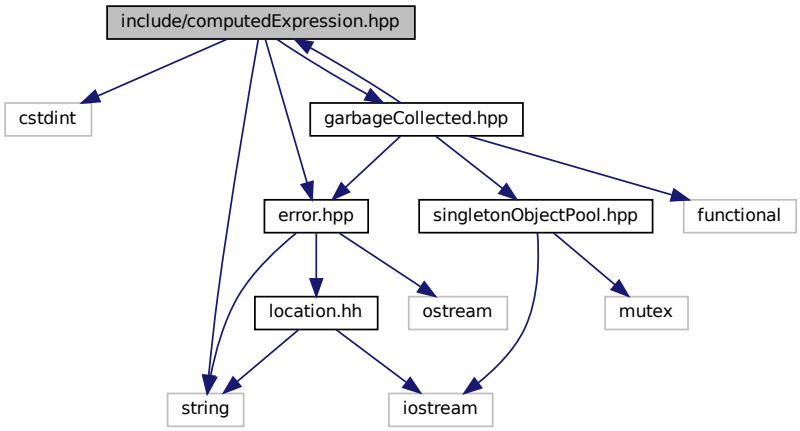
Declare the Tang::AstNodeUnary class.

6.14 include/computedExpression.hpp File Reference

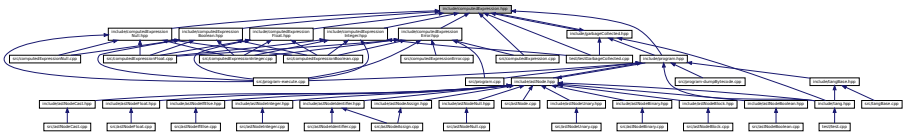
Declare the Tang::ComputedExpression base class.

```
#include <cstdint>
#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.

6.14.1 Detailed Description

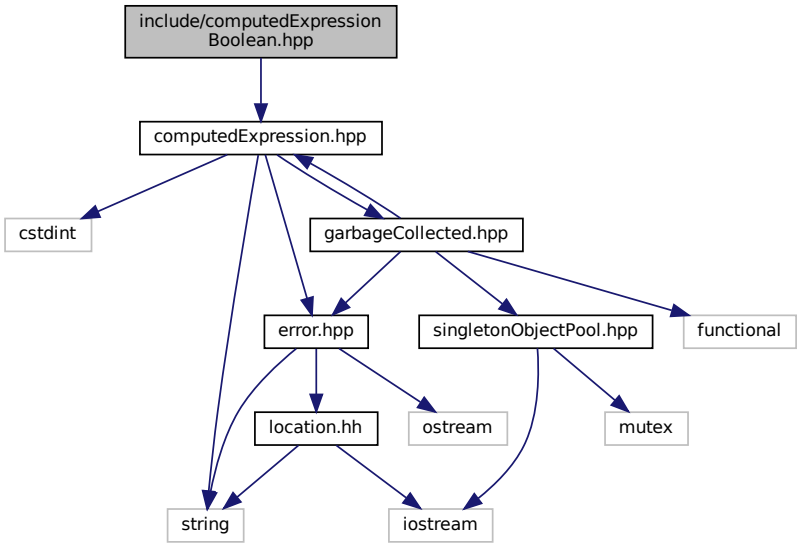
Declare the Tang::ComputedExpression base class.

6.15 include/computedExpressionBoolean.hpp File Reference

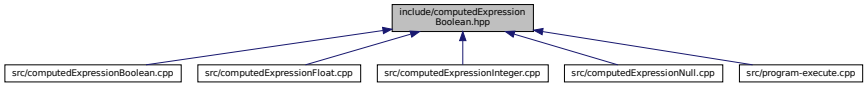
Declare the Tang::ComputedExpressionBoolean class.

```
#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.

6.15.1 Detailed Description

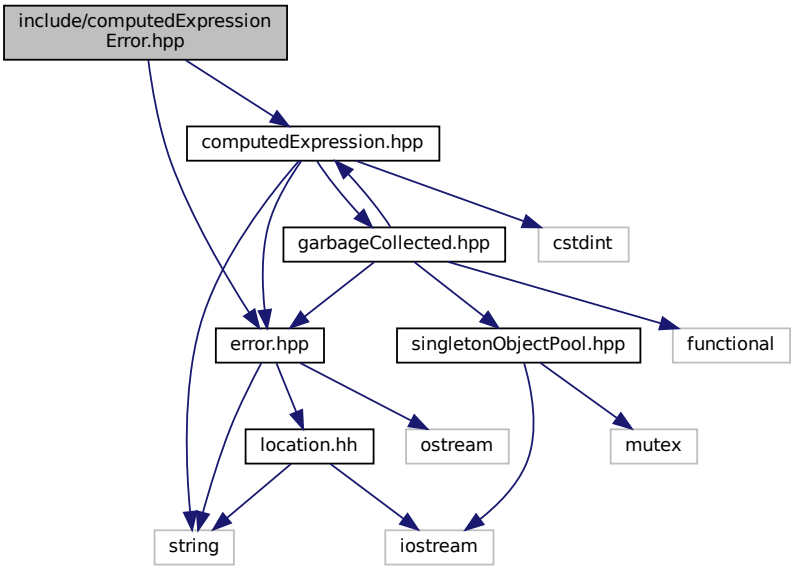
Declare the Tang::ComputedExpressionBoolean class.

6.16 include/computedExpressionError.hpp File Reference

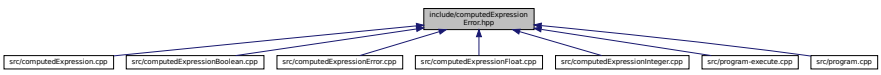
Declare the Tang::ComputedExpressionError class.

```
#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.

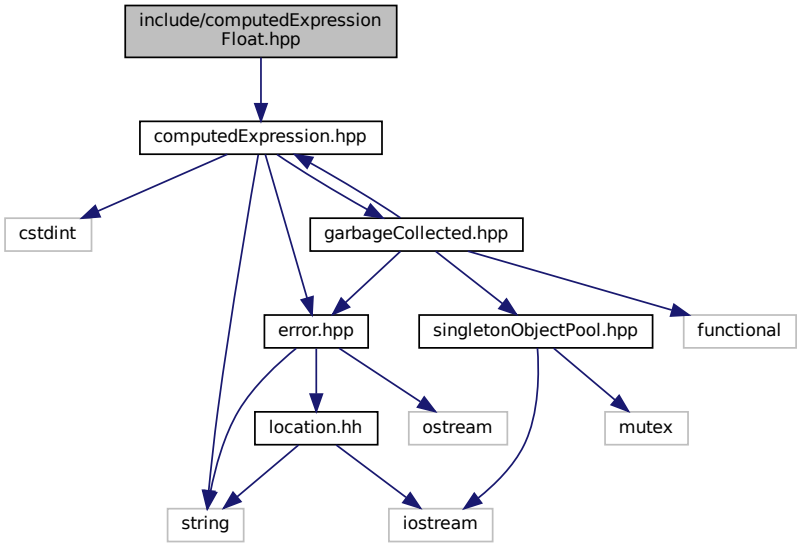
6.16.1 Detailed Description

Declare the Tang::ComputedExpressionError class.

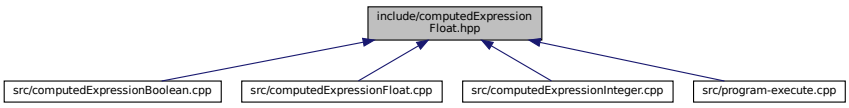
6.17 include/computedExpressionFloat.hpp File Reference

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.

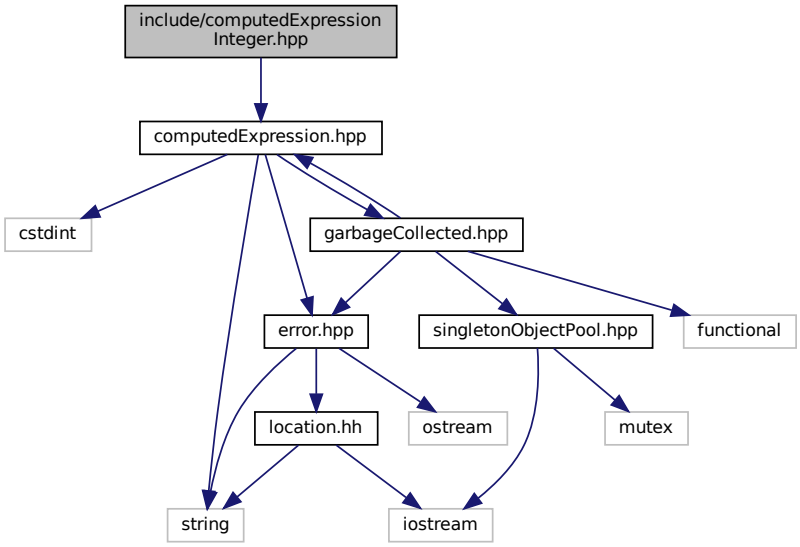
6.17.1 Detailed Description

Declare the Tang::ComputedExpressionFloat class.

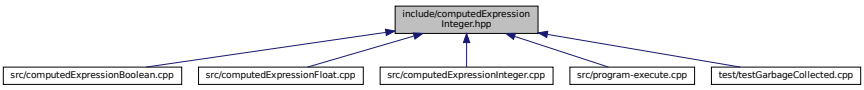
6.18 include/computedExpressionInteger.hpp File Reference

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.

6.18.1 Detailed Description

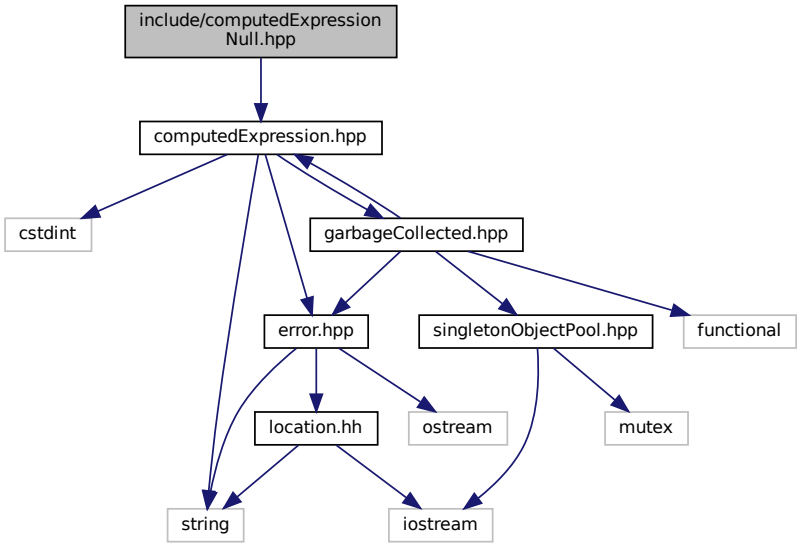
Declare the Tang::ComputedExpressionInteger class.

6.19 include/computedExpressionNull.hpp File Reference

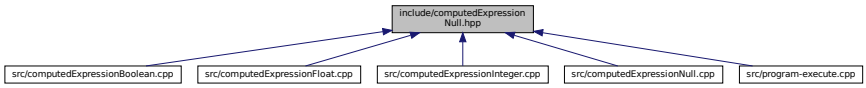
Declare the Tang::ComputedExpressionNull class.

```
#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.

6.19.1 Detailed Description

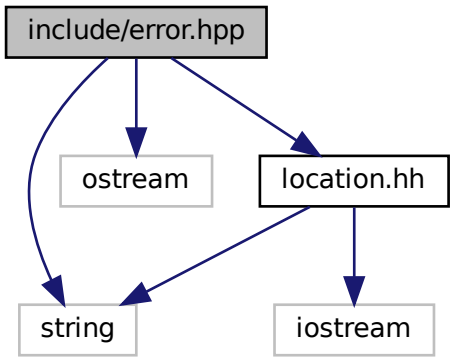
Declare the Tang::ComputedExpressionNull class.

6.20 include/error.hpp File Reference

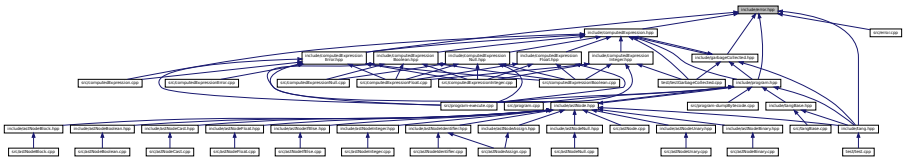
Declare the Tang::Error class used to describe syntax and runtime errors.

```
#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.

6.20.1 Detailed Description

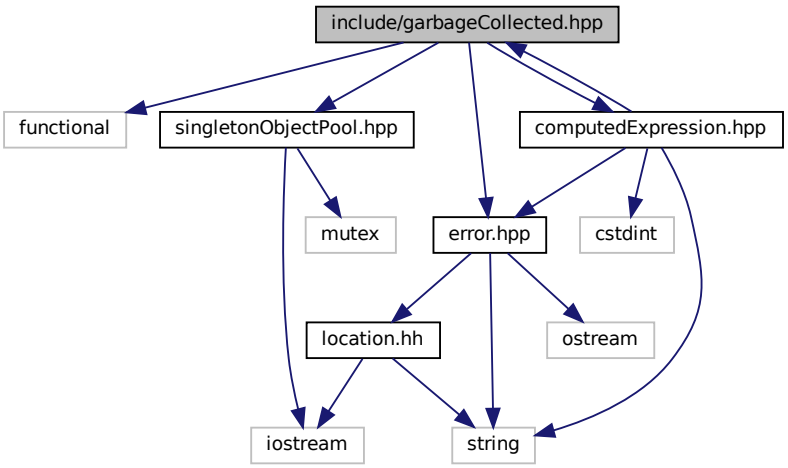
Declare the Tang::Error class used to describe syntax and runtime errors.

6.21 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.

6.21.1 Detailed Description

Declare the Tang::GarbageCollected class.

6.22 include/macros.hpp File Reference

Contains generic macros.

Macros

- #define TANG_UNUSED(x) x

  Instruct the compiler that a function argument will not be used so that it does not generate an error.

6.22.1 Detailed Description

Contains generic macros.

6.22.2 Macro Definition Documentation

6.22.2.1 TANG_UNUSED

```
#define TANG_UNUSED(
                x ) x
```

Instruct the compiler that a function argument will not be used so that it does not generate an error.

When defining a funcion, use the TANG_UNUSED() macro around any argument which is *not* used in the function, in order to squash any compiler warnings. e.g., void foo(int TANG_UNUSED(a)) {}

Parameters

x	The argument to be ignored.

6.23 include/opcode.hpp File Reference

Declare the Opcodes used in the Bytecode representation of a program.

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

Enumerations

- enum class Tang::Opcode {
  POP , PEEK , POKE , JMP ,
  JMPF_POP , NULLVAL , INTEGER , FLOAT ,
  BOOLEAN , ADD , SUBTRACT , MULTIPLY ,
  DIVIDE , MODULO , NEGATIVE , NOT ,
  LT , LTE , GT , GTE ,
  EQ , NEQ , CASTINTEGER , CASTFLOAT ,
  CASTBOOLEAN }

6.23.1 Detailed Description

Declare the Opcodes used in the Bytecode representation of a program.

6.23.2 Enumeration Type Documentation

6.23.2.1 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 #.
JMP	PC #: set pc to PC #.
JMPF_POP	PC #: pop val, if false, set pc to PC #.
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.

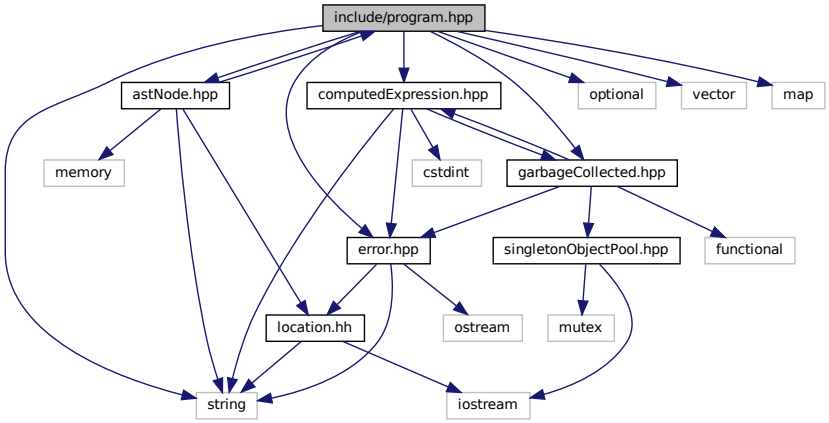
6.24 include/program.hpp File Reference

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.

6.24.1 Detailed Description

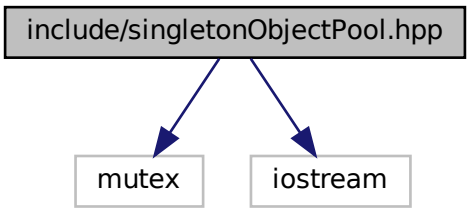
Declare the Tang::Program class used to compile and execute source code.

6.25 include/singletonObjectPool.hpp File Reference

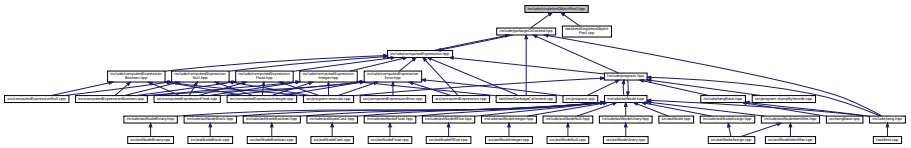
Declare the Tang::SingletonObjectPool class.

```
#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.

6.25.1 Detailed Description

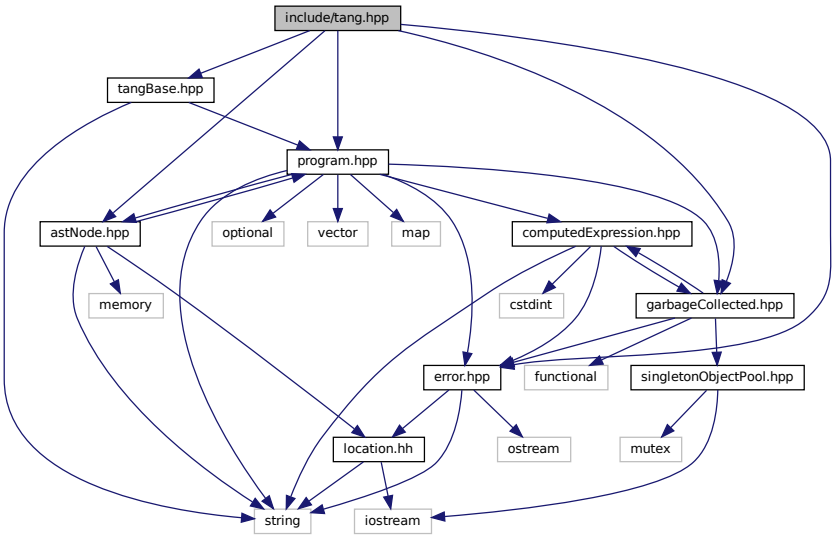
Declare the Tang::SingletonObjectPool class.

6.26 include/tang.hpp File Reference

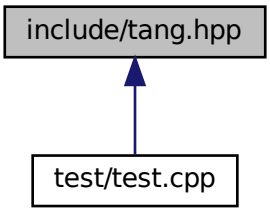
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

```
#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:



6.26.1 Detailed Description

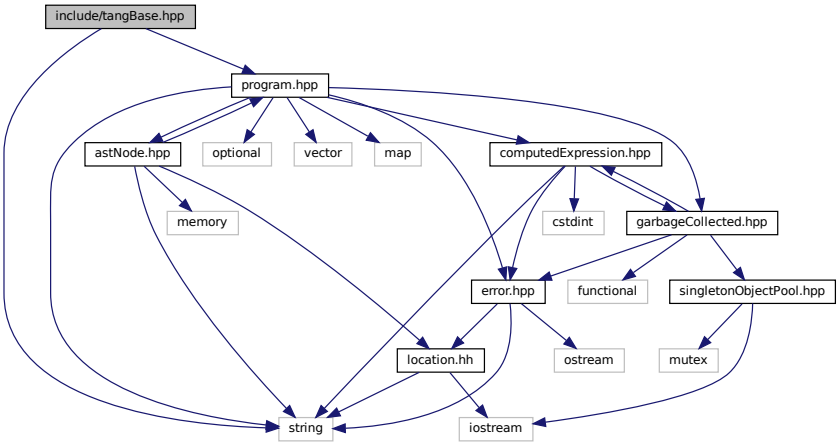
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

6.27 include/tangBase.hpp File Reference

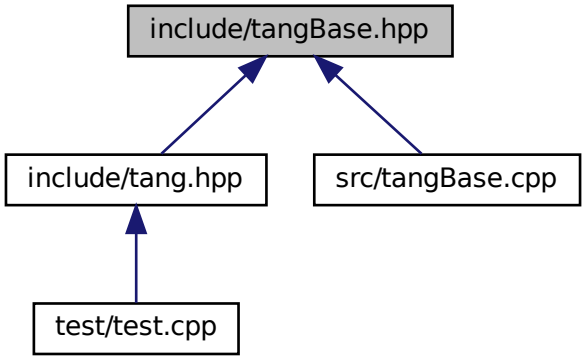
Declare the Tang::TangBase class used to interact with Tang.

```
#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.

6.27.1 Detailed Description

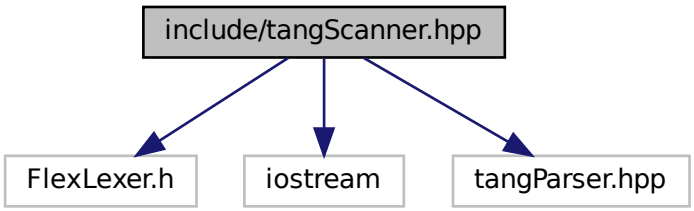
Declare the Tang::TangBase class used to interact with Tang.

6.28 include/tangScanner.hpp File Reference

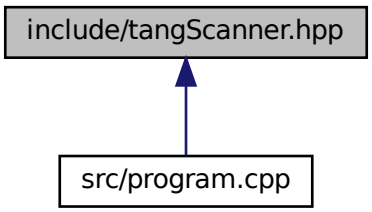
Declare the Tang::TangScanner used to tokenize a Tang script.

```
#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()

6.28.1 Detailed Description

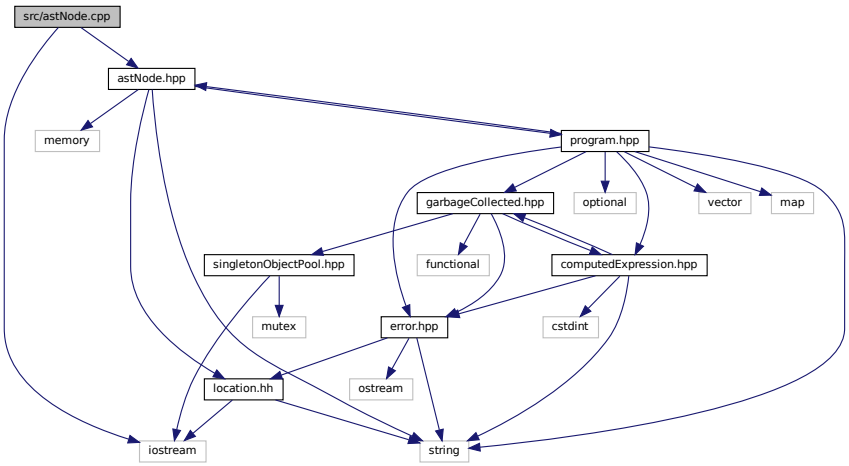
Declare the Tang::TangScanner used to tokenize a Tang script.

6.29 src/astNode.cpp File Reference

Define the Tang::AstNode class.

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

Include dependency graph for astNode.cpp:



6.29.1 Detailed Description

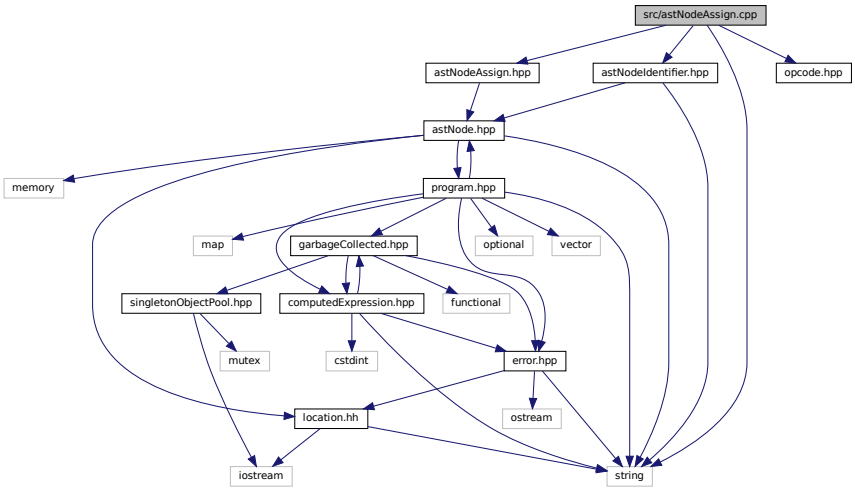
Define the Tang::AstNode class.

6.30 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:



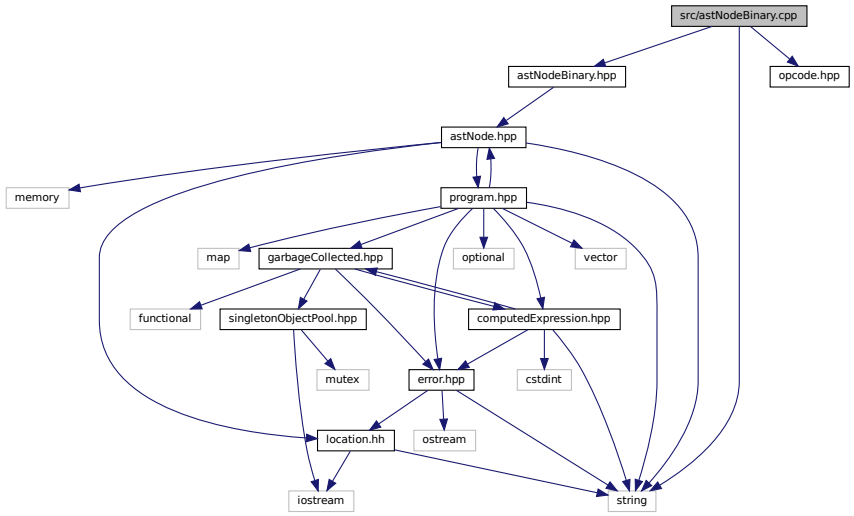
6.30.1 Detailed Description

Define the Tang::AstNodeAssign class.

6.31 src/astNodeBinary.cpp File Reference

Define the Tang::AstNodeBinary class.

```
#include <string>
#include "astNodeBinary.hpp"
#include "opcode.hpp"
```
Include dependency graph for astNodeBinary.cpp:



6.31.1 Detailed Description

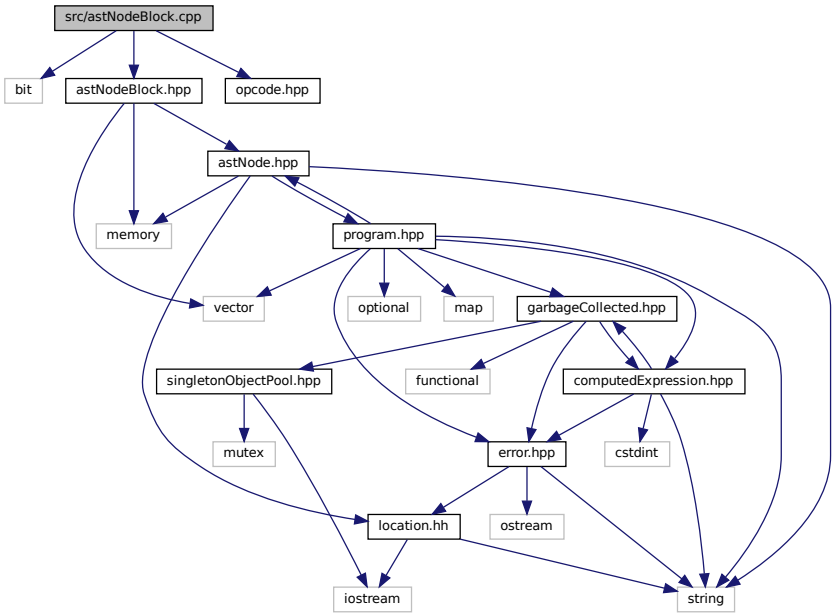
Define the Tang::AstNodeBinary class.

6.32 src/astNodeBlock.cpp File Reference

Define the Tang::AstNodeBlock class.

```
#include <bit>
#include "astNodeBlock.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeBlock.cpp:



6.32.1 Detailed Description

Define the Tang::AstNodeBlock class.

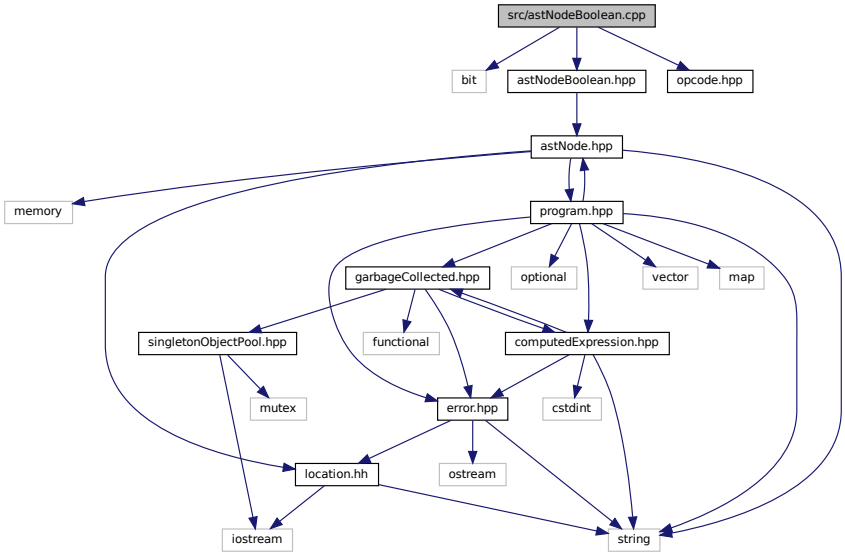
6.33 src/astNodeBoolean.cpp File Reference

Define the Tang::AstNodeBoolean class.

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

```
#include "opcode.hpp"
```

Include dependency graph for astNodeBoolean.cpp:



6.33.1 Detailed Description

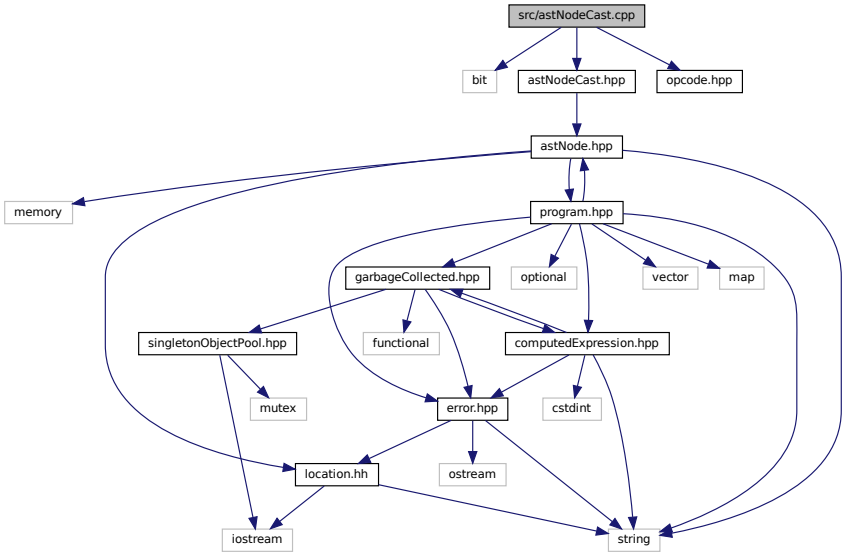
Define the Tang::AstNodeBoolean class.

6.34 src/astNodeCast.cpp File Reference

Define the Tang::AstNodeCast class.

```
#include <bit>
#include "astNodeCast.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeCast.cpp:



6.34.1 Detailed Description

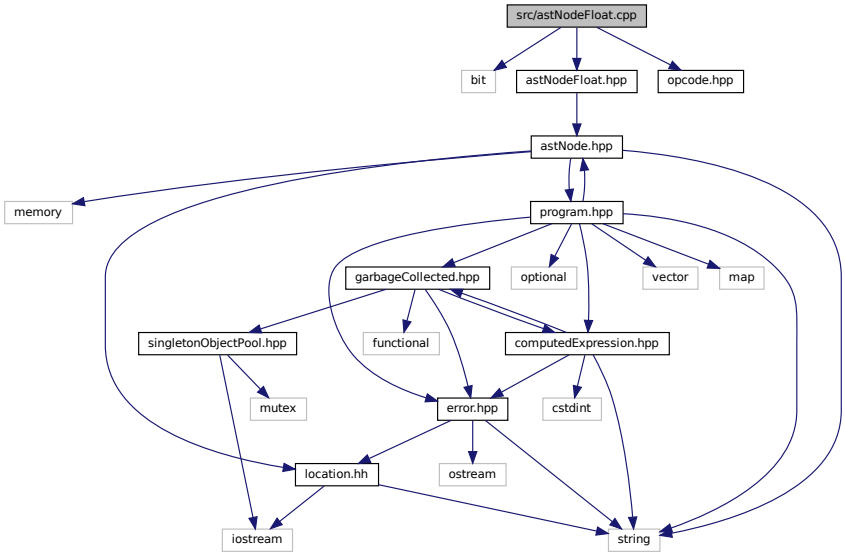
Define the Tang::AstNodeCast class.

6.35 src/astNodeFloat.cpp File Reference

Define the Tang::AstNodeFloat class.

```
#include <bit>
#include "astNodeFloat.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeFloat.cpp:



6.35.1 Detailed Description

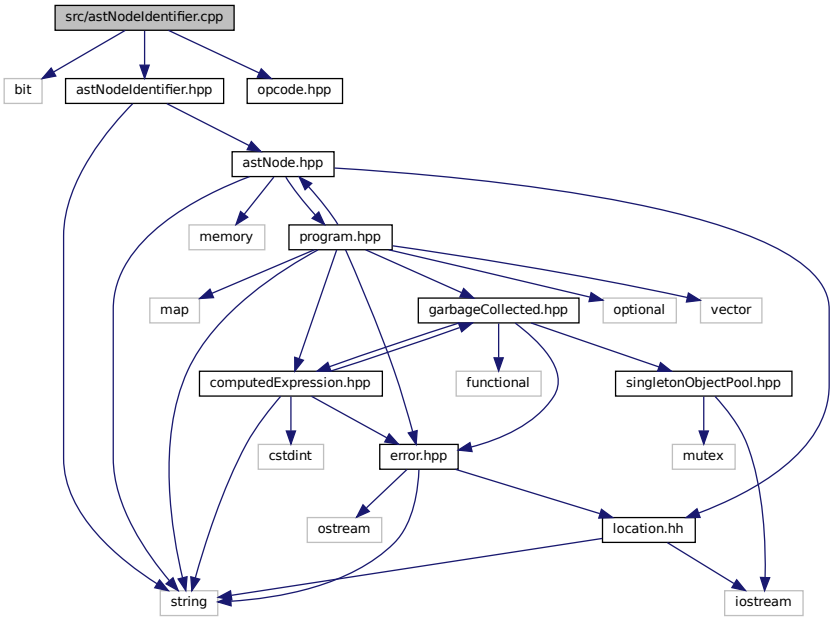
Define the Tang::AstNodeFloat class.

6.36 src/astNodeIdentifier.cpp File Reference

Define the Tang::AstNodeIdentifier class.

```
#include <bit>
#include "astNodeIdentifier.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeIdentifier.cpp:



6.36.1 Detailed Description

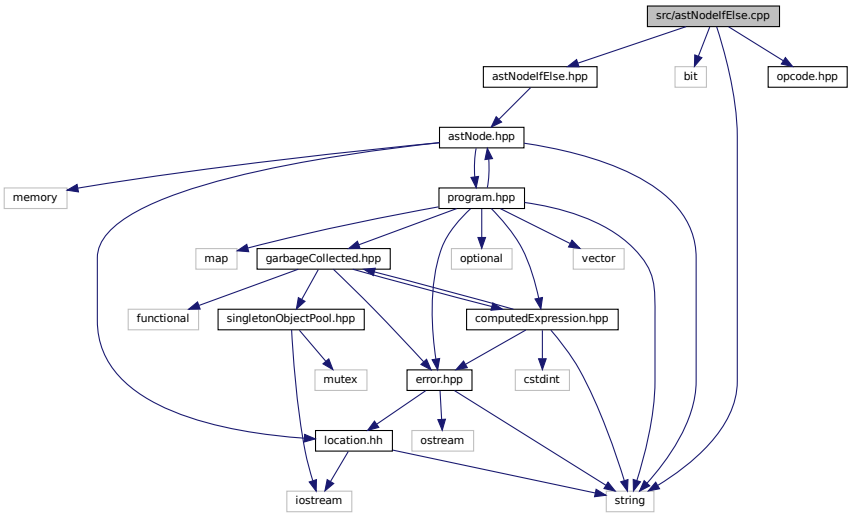
Define the Tang::AstNodeIdentifier class.

6.37 src/astNodeIfElse.cpp File Reference

Define the Tang::AstNodeIfElse class.

```
#include <string>
#include <bit>
#include "astNodeIfElse.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeIfElse.cpp:



6.37.1 Detailed Description

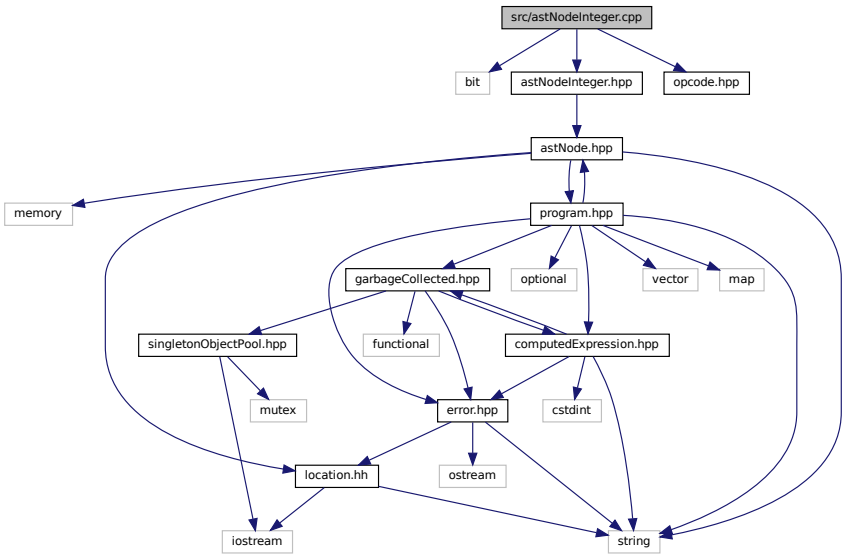
Define the Tang::AstNodeIfElse class.

6.38 src/astNodeInteger.cpp File Reference

Define the Tang::AstNodeInteger class.

```
#include <bit>
#include "astNodeInteger.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeInteger.cpp:



6.38.1 Detailed Description

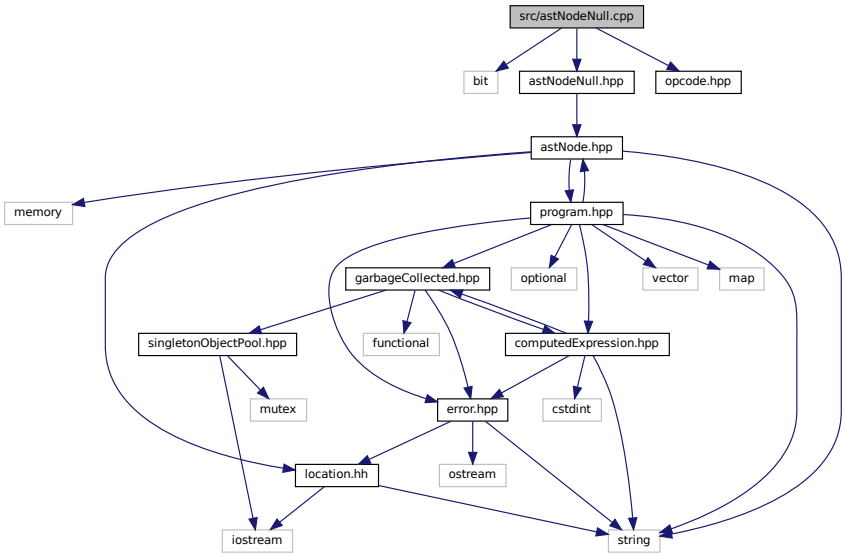
Define the Tang::AstNodeInteger class.

6.39 src/astNodeNull.cpp File Reference

Define the Tang::AstNodeNull class.

```
#include <bit>
#include "astNodeNull.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodeNull.cpp:



6.39.1 Detailed Description

Define the Tang::AstNodeNull class.

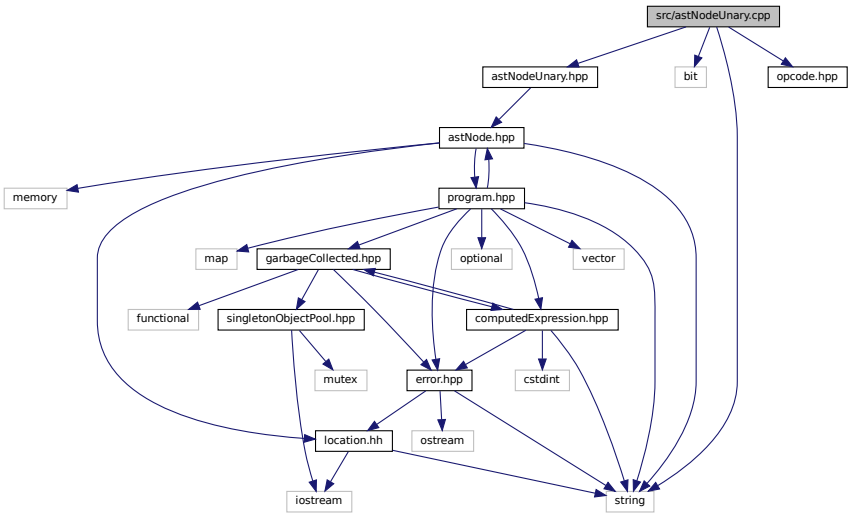
6.40 src/astNodeUnary.cpp File Reference

Define the Tang::AstNodeUnary class.

```
#include <string>
#include <bit>
#include "astNodeUnary.hpp"
```

```
#include "opcode.hpp"
```

Include dependency graph for astNodeUnary.cpp:



6.40.1 Detailed Description

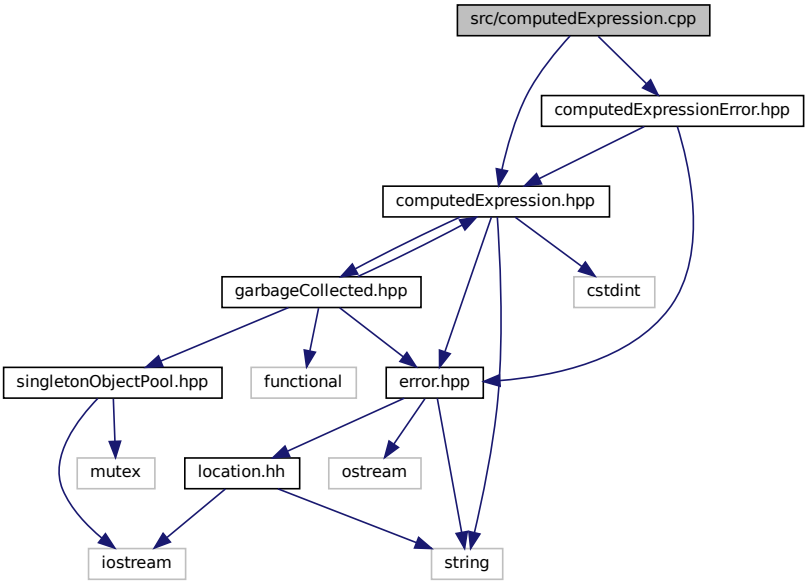
Define the Tang::AstNodeUnary class.

6.41 src/computedExpression.cpp File Reference

Define the Tang::ComputedExpression class.

```
#include "computedExpression.hpp"
#include "computedExpressionError.hpp"
```

Include dependency graph for computedExpression.cpp:



6.41.1 Detailed Description

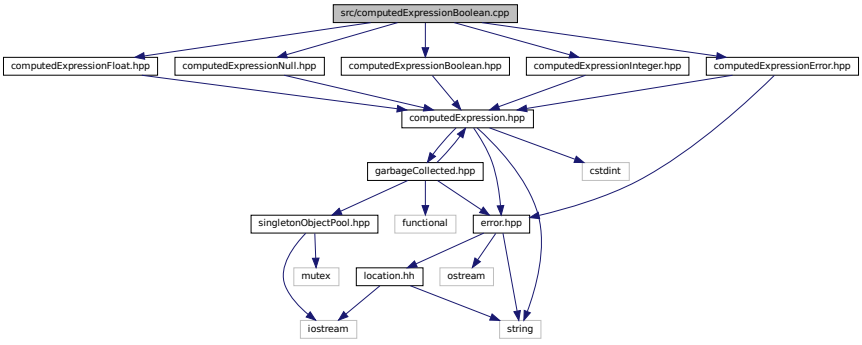
Define the Tang::ComputedExpression class.

6.42 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:



6.42.1 Detailed Description

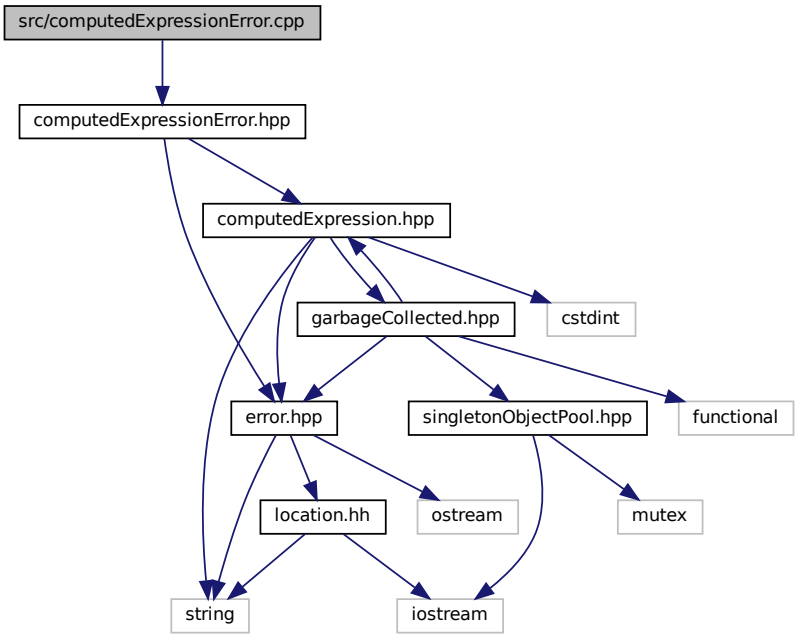
Define the Tang::ComputedExpressionBoolean class.

6.43 src/computedExpressionError.cpp File Reference

Define the Tang::ComputedExpressionError class.

```
#include "computedExpressionError.hpp"
```

Include dependency graph for computedExpressionError.cpp:



6.43.1 Detailed Description

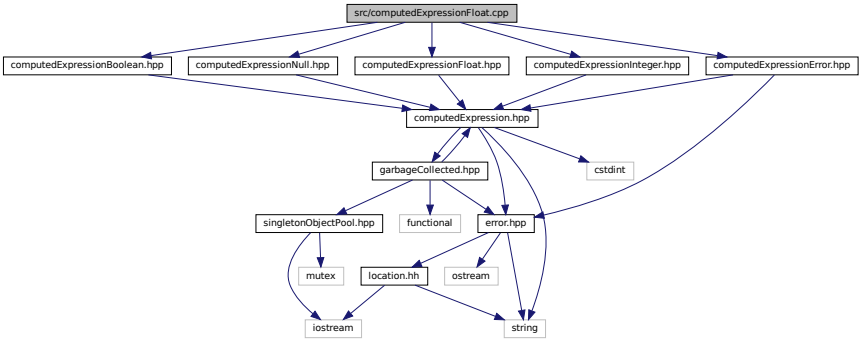
Define the Tang::ComputedExpressionError class.

6.44 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:



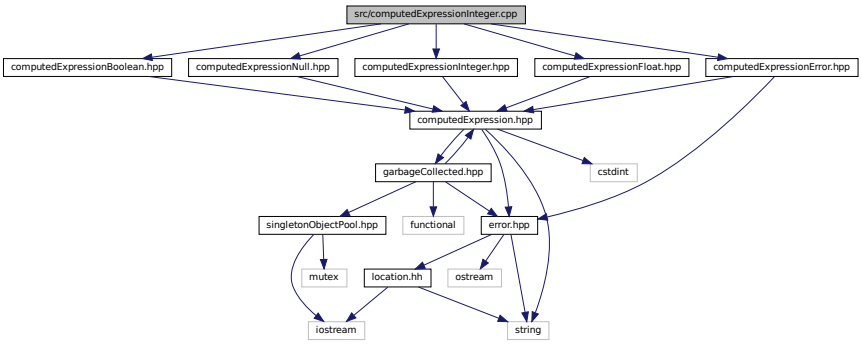
6.44.1 Detailed Description

Define the Tang::ComputedExpressionFloat class.

6.45 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:



6.45.1 Detailed Description

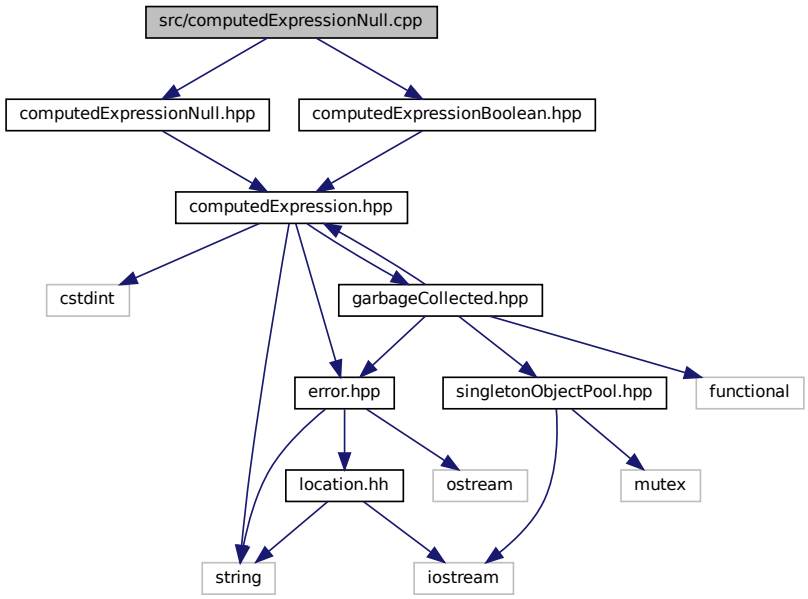
Define the Tang::ComputedExpressionInteger class.

6.46 src/computedExpressionNull.cpp File Reference

Define the Tang::ComputedExpressionNull class.

```
#include "computedExpressionNull.hpp"
#include "computedExpressionBoolean.hpp"
```

Include dependency graph for computedExpressionNull.cpp:



6.46.1 Detailed Description

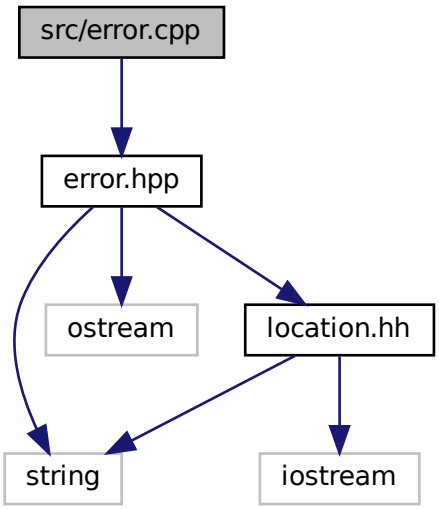
Define the Tang::ComputedExpressionNull class.

6.47 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)

6.47.1 Detailed Description

Define the Tang::Error class.

6.47.2 Function Documentation

6.47.2.1 operator<<()

```
std::ostream& Tang::operator<< (
            std::ostream & out,
            const Error & error )
```

Parameters

out	The output stream.
error	The Error object.

Returns

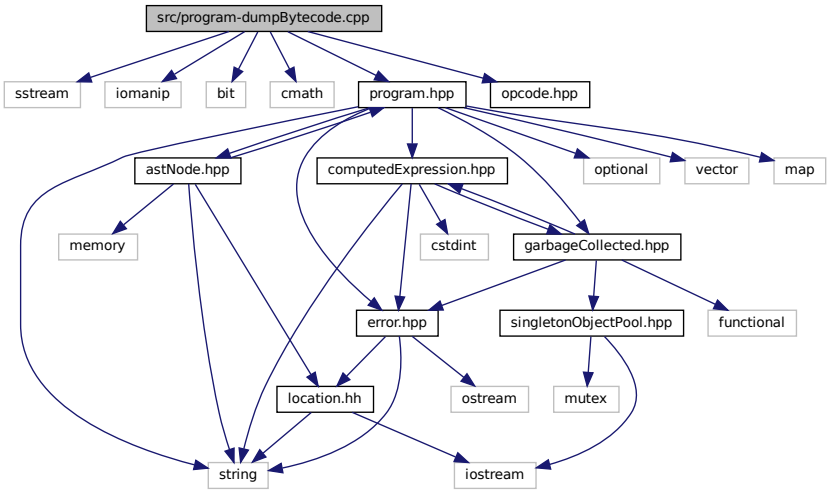
The output stream.

6.48 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.

6.48.1 Detailed Description

Define the Tang::Program::dumpBytecode method.

6.48.2 Macro Definition Documentation

6.48.2.1 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

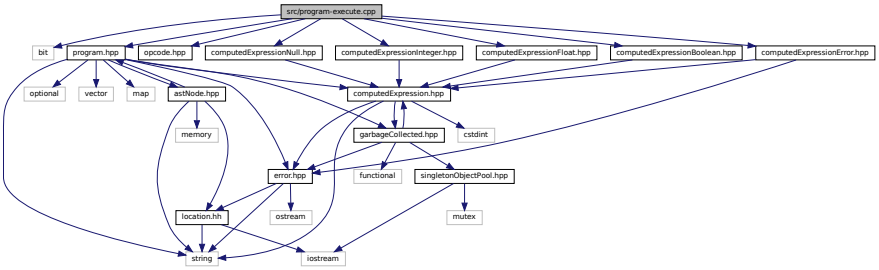
x	The number of additional vector entries that should exist.

6.49 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.

6.49.1 Detailed Description

Define the Tang::Program::execute method.

6.49.2 Macro Definition Documentation

6.49.2.1 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.

6.49.2.2 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

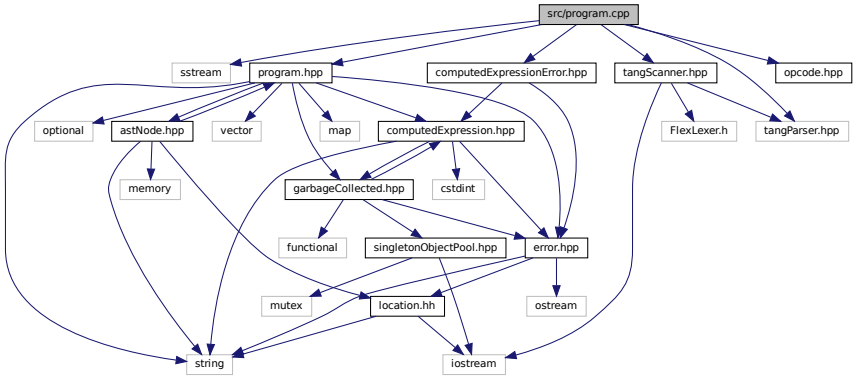
x	The number of entries that should exist in the stack.

6.50 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:



6.50.1 Detailed Description

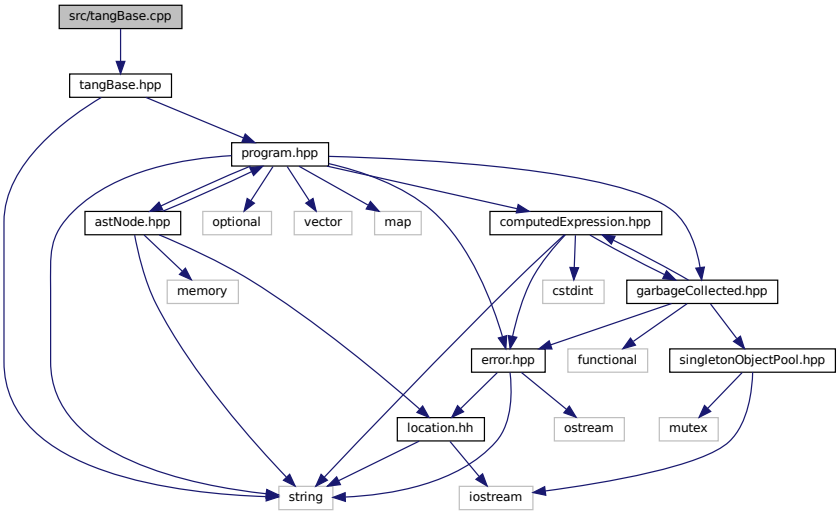
Define the Tang::Program class.

6.51 src/tangBase.cpp File Reference

Define the Tang::TangBase class.

```
#include "tangBase.hpp"
```

Include dependency graph for tangBase.cpp:



6.51.1 Detailed Description

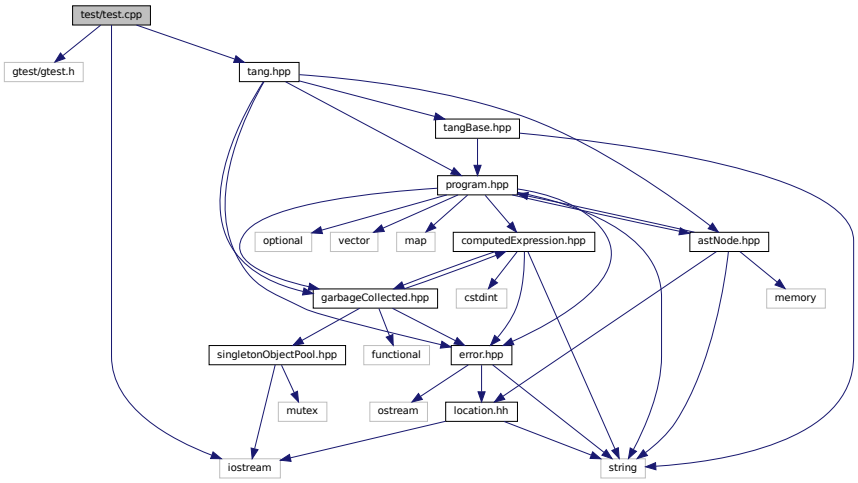
Define the Tang::TangBase class.

6.52 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)
- **TEST** (Assign, IfElse)
- int **main** (int argc, char ∗∗argv)

6.52.1 Detailed Description

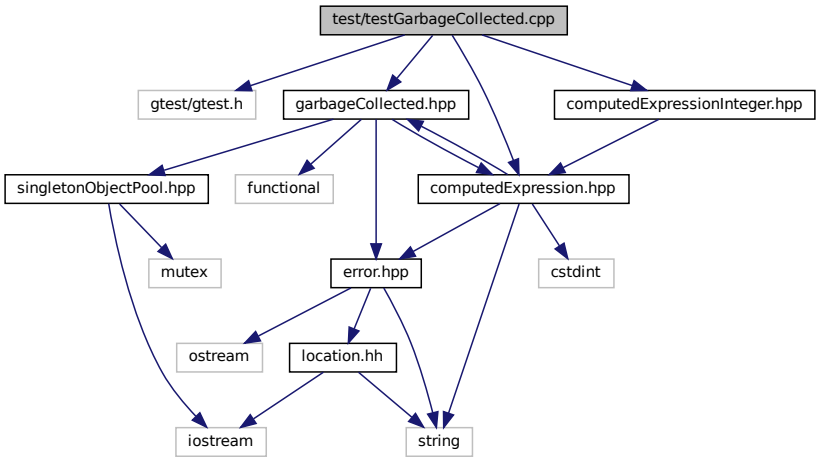
Test the general language behaviors.

6.53 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)

6.53.1 Detailed Description

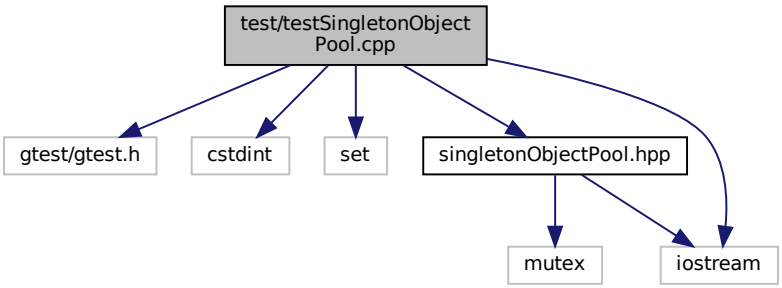
Test the generic behavior of the Tang::GarbageCollected class.

6.54 test/testSingletonObjectPool.cpp File Reference

Test the generic behavior of the Tang::SingletonObjectPool class.

```
#include <gtest/gtest.h>
#include <cstdint>
#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)

6.54.1 Detailed Description

Test the generic behavior of the Tang::SingletonObjectPool class.

Index

__add
Tang::ComputedExpression, 50
Tang::ComputedExpressionBoolean, 59
Tang::ComputedExpressionError, 67
Tang::ComputedExpressionFloat, 76
Tang::ComputedExpressionInteger, 85
Tang::ComputedExpressionNull, 94
__boolean
Tang::ComputedExpression, 50
Tang::ComputedExpressionBoolean, 59
Tang::ComputedExpressionError, 68
Tang::ComputedExpressionFloat, 77
Tang::ComputedExpressionInteger, 86
Tang::ComputedExpressionNull, 94
__divide
Tang::ComputedExpression, 51
Tang::ComputedExpressionBoolean, 59
Tang::ComputedExpressionError, 68
Tang::ComputedExpressionFloat, 77
Tang::ComputedExpressionInteger, 86
Tang::ComputedExpressionNull, 94
__equal
Tang::ComputedExpression, 51
Tang::ComputedExpressionBoolean, 60
Tang::ComputedExpressionError, 68
Tang::ComputedExpressionFloat, 77
Tang::ComputedExpressionInteger, 86
Tang::ComputedExpressionNull, 95
__float
Tang::ComputedExpression, 51
Tang::ComputedExpressionBoolean, 60
Tang::ComputedExpressionError, 69
Tang::ComputedExpressionFloat, 78
Tang::ComputedExpressionInteger, 87
Tang::ComputedExpressionNull, 95
__integer
Tang::ComputedExpression, 52
Tang::ComputedExpressionBoolean, 60
Tang::ComputedExpressionError, 69
Tang::ComputedExpressionFloat, 78
Tang::ComputedExpressionInteger, 87
Tang::ComputedExpressionNull, 95
__lessThan
Tang::ComputedExpression, 52
Tang::ComputedExpressionBoolean, 61
Tang::ComputedExpressionError, 69
Tang::ComputedExpressionFloat, 78
Tang::ComputedExpressionInteger, 87
Tang::ComputedExpressionNull, 96
__modulo
Tang::ComputedExpression, 52
Tang::ComputedExpressionBoolean, 61
Tang::ComputedExpressionError, 70
Tang::ComputedExpressionFloat, 79
Tang::ComputedExpressionInteger, 88
Tang::ComputedExpressionNull, 96
__multiply
Tang::ComputedExpression, 53
Tang::ComputedExpressionBoolean, 61
Tang::ComputedExpressionError, 70
Tang::ComputedExpressionFloat, 79
Tang::ComputedExpressionInteger, 88
Tang::ComputedExpressionNull, 97
__negative
Tang::ComputedExpression, 53
Tang::ComputedExpressionBoolean, 62
Tang::ComputedExpressionError, 70
Tang::ComputedExpressionFloat, 79
Tang::ComputedExpressionInteger, 88
Tang::ComputedExpressionNull, 97
__not
Tang::ComputedExpression, 53
Tang::ComputedExpressionBoolean, 62
Tang::ComputedExpressionError, 71
Tang::ComputedExpressionFloat, 80
Tang::ComputedExpressionInteger, 89
Tang::ComputedExpressionNull, 97
__subtract
Tang::ComputedExpression, 54
Tang::ComputedExpressionBoolean, 62
Tang::ComputedExpressionError, 71
Tang::ComputedExpressionFloat, 80
Tang::ComputedExpressionInteger, 89
Tang::ComputedExpressionNull, 97
~GarbageCollected
Tang::GarbageCollected, 106

ADD
opcode.hpp, 156
Add
Tang::AstNodeBinary, 20
addBytecode
Tang::Program, 124
AstNode
Tang::AstNode, 14
AstNodeAssign
Tang::AstNodeAssign, 17
AstNodeBinary
Tang::AstNodeBinary, 20

AstNodeBlock
Tang::AstNodeBlock, 23
AstNodeBoolean
Tang::AstNodeBoolean, 27
AstNodeCast
Tang::AstNodeCast, 30
AstNodeFloat
Tang::AstNodeFloat, 33
AstNodeIdentifier
Tang::AstNodeIdentifier, 36
AstNodeIfElse
Tang::AstNodeIfElse, 39
AstNodeInteger
Tang::AstNodeInteger, 42
AstNodeNull
Tang::AstNodeNull, 45
AstNodeUnary
Tang::AstNodeUnary, 47

BOOLEAN
opcode.hpp, 156
Boolean
Tang::AstNodeCast, 30
build/generated/location.hh, 133

CASTBOOLEAN
opcode.hpp, 156
CASTFLOAT
opcode.hpp, 156
CASTINTEGER
opcode.hpp, 156
CodeType
Tang::Program, 123
compileIdentifiers
Tang::AstNode, 14
Tang::AstNodeAssign, 17
Tang::AstNodeBinary, 21
Tang::AstNodeBlock, 23
Tang::AstNodeBoolean, 27
Tang::AstNodeCast, 31
Tang::AstNodeFloat, 33
Tang::AstNodeIdentifier, 36
Tang::AstNodeIfElse, 40
Tang::AstNodeInteger, 42
Tang::AstNodeNull, 45
Tang::AstNodeUnary, 48
compileScript
Tang::TangBase, 128
ComputedExpressionBoolean
Tang::ComputedExpressionBoolean, 58
ComputedExpressionError
Tang::ComputedExpressionError, 67
ComputedExpressionFloat
Tang::ComputedExpressionFloat, 76
ComputedExpressionInteger
Tang::ComputedExpressionInteger, 85

DIVIDE
opcode.hpp, 156
Divide
Tang::AstNodeBinary, 20
dump
Tang::ComputedExpression, 54
Tang::ComputedExpressionBoolean, 63
Tang::ComputedExpressionError, 71
Tang::ComputedExpressionFloat, 80
Tang::ComputedExpressionInteger, 89
Tang::ComputedExpressionNull, 98
dumpBytecode
Tang::Program, 124
DUMPPROGRAMCHECK
program-dumpBytecode.cpp, 177

EQ
opcode.hpp, 156
Equal
Tang::AstNodeBinary, 20
Error
Tang::Error, 102
error.cpp
operator<<, 176
execute
Tang::Program, 124
EXECUTEPROGRAMCHECK
program-execute.cpp, 179

FLOAT
opcode.hpp, 156
Float
Tang::AstNodeCast, 30

GarbageCollected
Tang::GarbageCollected, 105, 106
get
Tang::SingletonObjectPool< T >, 127
get_next_token
Tang::TangScanner, 130
getAst
Tang::Program, 125
getBytecode
Tang::Program, 125
getCode
Tang::Program, 125
getInstance
Tang::SingletonObjectPool< T >, 127
getResult
Tang::Program, 125
GreaterThan
Tang::AstNodeBinary, 20
GreaterThanEqual
Tang::AstNodeBinary, 20
GT
opcode.hpp, 156
GTE
opcode.hpp, 156

include/astNode.hpp, 135
include/astNodeAssign.hpp, 136

include/astNodeBinary.hpp, 137
include/astNodeBlock.hpp, 138
include/astNodeBoolean.hpp, 139
include/astNodeCast.hpp, 140
include/astNodeFloat.hpp, 141
include/astNodeIdentifier.hpp, 142
include/astNodeIfElse.hpp, 143
include/astNodeInteger.hpp, 144
include/astNodeNull.hpp, 145
include/astNodeUnary.hpp, 146
include/computedExpression.hpp, 147
include/computedExpressionBoolean.hpp, 148
include/computedExpressionError.hpp, 149
include/computedExpressionFloat.hpp, 150
include/computedExpressionInteger.hpp, 151
include/computedExpressionNull.hpp, 152
include/error.hpp, 153
include/garbageCollected.hpp, 154
include/macros.hpp, 154
include/opcode.hpp, 155
include/program.hpp, 156
include/singletonObjectPool.hpp, 158
include/tang.hpp, 159
include/tangBase.hpp, 160
include/tangScanner.hpp, 161
INTEGER
    opcode.hpp, 156
Integer
    Tang::AstNodeCast, 30
is_equal
    Tang::ComputedExpression, 54–56
    Tang::ComputedExpressionBoolean, 63, 64
    Tang::ComputedExpressionError, 72, 73
    Tang::ComputedExpressionFloat, 81, 82
    Tang::ComputedExpressionInteger, 90, 91
    Tang::ComputedExpressionNull, 98, 99

JMP
    opcode.hpp, 156
JMPF_POP
    opcode.hpp, 156

LessThan
    Tang::AstNodeBinary, 20
LessThanEqual
    Tang::AstNodeBinary, 20
location.hh
    operator<<, 134, 135
LT
    opcode.hpp, 156
LTE
    opcode.hpp, 156

macros.hpp
    TANG_UNUSED, 155
make
    Tang::GarbageCollected, 106
makeCopy
    Tang::ComputedExpression, 56
    Tang::ComputedExpressionBoolean, 65
    Tang::ComputedExpressionError, 74
    Tang::ComputedExpressionFloat, 83
    Tang::ComputedExpressionInteger, 92
    Tang::ComputedExpressionNull, 100
MODULO
    opcode.hpp, 156
Modulo
    Tang::AstNodeBinary, 20
MULTIPLY
    opcode.hpp, 156
Multiply
    Tang::AstNodeBinary, 20

NEGATIVE
    opcode.hpp, 156
Negative
    Tang::AstNodeUnary, 47
NEQ
    opcode.hpp, 156
NOT
    opcode.hpp, 156
Not
    Tang::AstNodeUnary, 47
NotEqual
    Tang::AstNodeBinary, 20
NULLVAL
    opcode.hpp, 156

Opcode
    opcode.hpp, 156
opcode.hpp
    ADD, 156
    BOOLEAN, 156
    CASTBOOLEAN, 156
    CASTFLOAT, 156
    CASTINTEGER, 156
    DIVIDE, 156
    EQ, 156
    FLOAT, 156
    GT, 156
    GTE, 156
    INTEGER, 156
    JMP, 156
    JMPF_POP, 156
    LT, 156
    LTE, 156
    MODULO, 156
    MULTIPLY, 156
    NEGATIVE, 156
    NEQ, 156
    NOT, 156
    NULLVAL, 156
    Opcode, 156
    PEEK, 156
    POKE, 156
    POP, 156
    SUBTRACT, 156
Operation

Tang::AstNodeBinary, 20
Operator
Tang::AstNodeUnary, 47
operator!
Tang::GarbageCollected, 107
operator!=
Tang::GarbageCollected, 107
operator<
Tang::GarbageCollected, 112
operator<<
error.cpp, 176
location.hh, 134, 135
Tang::Error, 102
Tang::GarbageCollected, 118
operator<=
Tang::GarbageCollected, 112
operator>
Tang::GarbageCollected, 116
operator>=
Tang::GarbageCollected, 116
operator*
Tang::GarbageCollected, 108, 109
operator+
Tang::GarbageCollected, 109
operator-
Tang::GarbageCollected, 110
operator->
Tang::GarbageCollected, 111
operator/
Tang::GarbageCollected, 111
operator=
Tang::GarbageCollected, 113
operator==
Tang::GarbageCollected, 114–116
operator%
Tang::GarbageCollected, 108

PEEK
opcode.hpp, 156
POKE
opcode.hpp, 156
POP
opcode.hpp, 156
Program
Tang::Program, 123
program-dumpBytecode.cpp
DUMPPROGRAMCHECK, 177
program-execute.cpp
EXECUTEPROGRAMCHECK, 179
STACKCHECK, 179

recycle
Tang::SingletonObjectPool< T >, 127

Script
Tang::Program, 123
setJumpTarget
Tang::Program, 126
src/astNode.cpp, 162
src/astNodeAssign.cpp, 162
src/astNodeBinary.cpp, 163
src/astNodeBlock.cpp, 164
src/astNodeBoolean.cpp, 164
src/astNodeCast.cpp, 165
src/astNodeFloat.cpp, 166
src/astNodeIdentifier.cpp, 167
src/astNodeIfElse.cpp, 168
src/astNodeInteger.cpp, 169
src/astNodeNull.cpp, 170
src/astNodeUnary.cpp, 170
src/computedExpression.cpp, 171
src/computedExpressionBoolean.cpp, 172
src/computedExpressionError.cpp, 173
src/computedExpressionFloat.cpp, 173
src/computedExpressionInteger.cpp, 174
src/computedExpressionNull.cpp, 175
src/error.cpp, 175
src/program-dumpBytecode.cpp, 177
src/program-execute.cpp, 178
src/program.cpp, 179
src/tangBase.cpp, 180
STACKCHECK
program-execute.cpp, 179
SUBTRACT
opcode.hpp, 156
Subtract
Tang::AstNodeBinary, 20

Tang::AstNode, 11
AstNode, 14
compileIdentifiers, 14
Tang::AstNodeAssign, 15
AstNodeAssign, 17
compileIdentifiers, 17
Tang::AstNodeBinary, 18
Add, 20
AstNodeBinary, 20
compileIdentifiers, 21
Divide, 20
Equal, 20
GreaterThan, 20
GreaterThanEqual, 20
LessThan, 20
LessThanEqual, 20
Modulo, 20
Multiply, 20
NotEqual, 20
Operation, 20
Subtract, 20
Tang::AstNodeBlock, 21
AstNodeBlock, 23
compileIdentifiers, 23
Tang::AstNodeBoolean, 25
AstNodeBoolean, 27
compileIdentifiers, 27
Tang::AstNodeCast, 28
AstNodeCast, 30
Boolean, 30

compileIdentifiers, 31
Float, 30
Integer, 30
Type, 30
Tang::AstNodeFloat, 31
AstNodeFloat, 33
compileIdentifiers, 33
Tang::AstNodeIdentifier, 34
AstNodeIdentifier, 36
compileIdentifiers, 36
Tang::AstNodeIfElse, 37
AstNodeIfElse, 39
compileIdentifiers, 40
Tang::AstNodeInteger, 40
AstNodeInteger, 42
compileIdentifiers, 42
Tang::AstNodeNull, 43
AstNodeNull, 45
compileIdentifiers, 45
Tang::AstNodeUnary, 45
AstNodeUnary, 47
compileIdentifiers, 48
Negative, 47
Not, 47
Operator, 47
Tang::ComputedExpression, 48
__add, 50
__boolean, 50
__divide, 51
__equal, 51
__float, 51
__integer, 52
__lessThan, 52
__modulo, 52
__multiply, 53
__negative, 53
__not, 53
__subtract, 54
dump, 54
is_equal, 54–56
makeCopy, 56
Tang::ComputedExpressionBoolean, 57
__add, 59
__boolean, 59
__divide, 59
__equal, 60
__float, 60
__integer, 60
__lessThan, 61
__modulo, 61
__multiply, 61
__negative, 62
__not, 62
__subtract, 62
ComputedExpressionBoolean, 58
dump, 63
is_equal, 63, 64
makeCopy, 65
Tang::ComputedExpressionError, 65
__add, 67
__boolean, 68
__divide, 68
__equal, 68
__float, 69
__integer, 69
__lessThan, 69
__modulo, 70
__multiply, 70
__negative, 70
__not, 71
__subtract, 71
ComputedExpressionError, 67
dump, 71
is_equal, 72, 73
makeCopy, 74
Tang::ComputedExpressionFloat, 74
__add, 76
__boolean, 77
__divide, 77
__equal, 77
__float, 78
__integer, 78
__lessThan, 78
__modulo, 79
__multiply, 79
__negative, 79
__not, 80
__subtract, 80
ComputedExpressionFloat, 76
dump, 80
is_equal, 81, 82
makeCopy, 83
Tang::ComputedExpressionInteger, 83
__add, 85
__boolean, 86
__divide, 86
__equal, 86
__float, 87
__integer, 87
__lessThan, 87
__modulo, 88
__multiply, 88
__negative, 88
__not, 89
__subtract, 89
ComputedExpressionInteger, 85
dump, 89
is_equal, 90, 91
makeCopy, 92
Tang::ComputedExpressionNull, 92
__add, 94
__boolean, 94
__divide, 94
__equal, 95
__float, 95
__integer, 95

__lessThan, 96
__modulo, 96
__multiply, 97
__negative, 97
__not, 97
__subtract, 97
dump, 98
is_equal, 98, 99
makeCopy, 100
Tang::Error, 100
Error, 102
operator<<, 102
Tang::GarbageCollected, 103
~GarbageCollected, 106
GarbageCollected, 105, 106
make, 106
operator!, 107
operator!=, 107
operator<, 112
operator<<, 118
operator<=, 112
operator>, 116
operator>=, 116
operator*, 108, 109
operator+, 109
operator-, 110
operator->, 111
operator/, 111
operator=, 113
operator==, 114–116
operator%, 108
Tang::location, 119
Tang::position, 120
Tang::Program, 122
addBytecode, 124
CodeType, 123
dumpBytecode, 124
execute, 124
getAst, 125
getBytecode, 125
getCode, 125
getResult, 125
Program, 123
Script, 123
setJumpTarget, 126
Template, 123
Tang::SingletonObjectPool< T >, 126
get, 127
getInstance, 127
recycle, 127
Tang::TangBase, 128
compileScript, 128
TangBase, 128
Tang::TangScanner, 129
get_next_token, 130
TangScanner, 130
TANG_UNUSED
macros.hpp, 155
TangBase
Tang::TangBase, 128
TangScanner
Tang::TangScanner, 130
Template
Tang::Program, 123
test/test.cpp, 181
test/testGarbageCollected.cpp, 182
test/testSingletonObjectPool.cpp, 183
Type
Tang::AstNodeCast, 30