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 . . . 13
5.1.2 Constructor & Destructor Documentation . . . 13
5.1.2.1 AstNode() . . . 13
5.1.3 Member Function Documentation . . . 13
5.1.3.1 collectIdentifiers() . . . 13
5.1.3.2 collectStrings() . . . 14
5.1.3.3 compile() . . . 14
5.1.3.4 dump() . . . 15
5.2 Tang::AstNodeAssign Class Reference . . . 15
5.2.1 Detailed Description . . . 16
5.2.2 Constructor & Destructor Documentation . . . 16
5.2.2.1 AstNodeAssign() . . . 16
5.2.3 Member Function Documentation . . . 17
5.2.3.1 collectIdentifiers() . . . 17
5.2.3.2 collectStrings() . . . 17
5.2.3.3 compile() . . . 17
5.2.3.4 dump() . . . 18
5.3 Tang::AstNodeBinary Class Reference . . . 18
5.3.1 Detailed Description . . . 20
5.3.2 Member Enumeration Documentation . . . 20
5.3.2.1 Operation . . . 20
5.3.3 Constructor & Destructor Documentation . . . 20
5.3.3.1 AstNodeBinary() . . . 20
5.3.4 Member Function Documentation . . . 21
5.3.4.1 collectIdentifiers() . . . 21
5.3.4.2 collectStrings() . . . 21
5.3.4.3 compile() . . . 21
5.3.4.4 dump() . . . 22

5.4 Tang::AstNodeBlock Class Reference . . . 22
5.4.1 Detailed Description . . . 23
5.4.2 Constructor & Destructor Documentation . . . 24
5.4.2.1 AstNodeBlock() . . . 24
5.4.3 Member Function Documentation . . . 24
5.4.3.1 collectIdentifiers() . . . 24
5.4.3.2 collectStrings() . . . 24
5.4.3.3 compile() . . . 25
5.4.3.4 dump() . . . 25
5.5 Tang::AstNodeBoolean Class Reference . . . 26
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 collectIdentifiers() . . . 27
5.5.3.2 collectStrings() . . . 27
5.5.3.3 compile() . . . 29
5.5.3.4 dump() . . . 29
5.6 Tang::AstNodeCast Class Reference . . . 30
5.6.1 Detailed Description . . . 31
5.6.2 Member Enumeration Documentation . . . 31
5.6.2.1 Type . . . 31
5.6.3 Constructor & Destructor Documentation . . . 31
5.6.3.1 AstNodeCast() . . . 31
5.6.4 Member Function Documentation . . . 32
5.6.4.1 collectIdentifiers() . . . 32
5.6.4.2 collectStrings() . . . 32
5.6.4.3 compile() . . . 32
5.6.4.4 dump() . . . 33
5.7 Tang::AstNodeDoWhile Class Reference . . . 33
5.7.1 Detailed Description . . . 34
5.7.2 Constructor & Destructor Documentation . . . 35
5.7.2.1 AstNodeDoWhile() . . . 35
5.7.3 Member Function Documentation . . . 35
5.7.3.1 collectIdentifiers() . . . 35
5.7.3.2 collectStrings() . . . 35
5.7.3.3 compile() . . . 36
5.7.3.4 dump() . . . 36
5.8 Tang::AstNodeFloat Class Reference . . . 37
5.8.1 Detailed Description . . . 38
5.8.2 Constructor & Destructor Documentation . . . 38
5.8.2.1 AstNodeFloat() . . . 38

5.8.3 Member Function Documentation . . . 38
5.8.3.1 collectIdentifiers() . . . 38
5.8.3.2 collectStrings() . . . 39
5.8.3.3 compile() . . . 39
5.8.3.4 dump() . . . 40
5.9 Tang::AstNodeFor Class Reference . . . 40
5.9.1 Detailed Description . . . 41
5.9.2 Constructor & Destructor Documentation . . . 41
5.9.2.1 AstNodeFor() . . . 41
5.9.3 Member Function Documentation . . . 42
5.9.3.1 collectIdentifiers() . . . 42
5.9.3.2 collectStrings() . . . 42
5.9.3.3 compile() . . . 42
5.9.3.4 dump() . . . 43
5.10 Tang::AstNodeIdentifier Class Reference . . . 44
5.10.1 Detailed Description . . . 45
5.10.2 Constructor & Destructor Documentation . . . 45
5.10.2.1 AstNodeIdentifier() . . . 45
5.10.3 Member Function Documentation . . . 45
5.10.3.1 collectIdentifiers() . . . 45
5.10.3.2 collectStrings() . . . 46
5.10.3.3 compile() . . . 46
5.10.3.4 dump() . . . 46
5.11 Tang::AstNodeIfElse Class Reference . . . 47
5.11.1 Detailed Description . . . 48
5.11.2 Constructor & Destructor Documentation . . . 48
5.11.2.1 AstNodeIfElse() [1/2] . . . 48
5.11.2.2 AstNodeIfElse() [2/2] . . . 48
5.11.3 Member Function Documentation . . . 49
5.11.3.1 collectIdentifiers() . . . 49
5.11.3.2 collectStrings() . . . 49
5.11.3.3 compile() . . . 49
5.11.3.4 dump() . . . 50
5.12 Tang::AstNodeInteger Class Reference . . . 51
5.12.1 Detailed Description . . . 52
5.12.2 Constructor & Destructor Documentation . . . 52
5.12.2.1 AstNodeInteger() . . . 52
5.12.3 Member Function Documentation . . . 52
5.12.3.1 collectIdentifiers() . . . 52
5.12.3.2 collectStrings() . . . 52
5.12.3.3 compile() . . . 53
5.12.3.4 dump() . . . 53

5.13 Tang::AstNodePrint Class Reference . . . 54
5.13.1 Detailed Description . . . 55
5.13.2 Member Enumeration Documentation . . . 55
5.13.2.1 Type . . . 55
5.13.3 Constructor & Destructor Documentation . . . 55
5.13.3.1 AstNodePrint() . . . 55
5.13.4 Member Function Documentation . . . 56
5.13.4.1 collectIdentifiers() . . . 56
5.13.4.2 collectStrings() . . . 56
5.13.4.3 compile() . . . 56
5.13.4.4 dump() . . . 57
5.14 Tang::AstNodeString Class Reference . . . 57
5.14.1 Detailed Description . . . 58
5.14.2 Constructor & Destructor Documentation . . . 59
5.14.2.1 AstNodeString() . . . 59
5.14.3 Member Function Documentation . . . 59
5.14.3.1 collectIdentifiers() . . . 59
5.14.3.2 collectStrings() . . . 59
5.14.3.3 compile() . . . 60
5.14.3.4 compileLiteral() . . . 60
5.14.3.5 dump() . . . 61
5.15 Tang::AstNodeUnary Class Reference . . . 61
5.15.1 Detailed Description . . . 63
5.15.2 Member Enumeration Documentation . . . 63
5.15.2.1 Operator . . . 63
5.15.3 Constructor & Destructor Documentation . . . 63
5.15.3.1 AstNodeUnary() . . . 63
5.15.4 Member Function Documentation . . . 63
5.15.4.1 collectIdentifiers() . . . 64
5.15.4.2 collectStrings() . . . 64
5.15.4.3 compile() . . . 64
5.15.4.4 dump() . . . 65
5.16 Tang::AstNodeWhile Class Reference . . . 65
5.16.1 Detailed Description . . . 66
5.16.2 Constructor & Destructor Documentation . . . 67
5.16.2.1 AstNodeWhile() . . . 67
5.16.3 Member Function Documentation . . . 67
5.16.3.1 collectIdentifiers() . . . 67
5.16.3.2 collectStrings() . . . 67
5.16.3.3 compile() . . . 68
5.16.3.4 dump() . . . 68
5.17 Tang::ComputedExpression Class Reference . . . 69

5.17.1 Detailed Description . . . 70
5.17.2 Member Function Documentation . . . 70
5.17.2.1 __add() . . . 70
5.17.2.2 __boolean() . . . 71
5.17.2.3 __divide() . . . 71
5.17.2.4 __equal() . . . 72
5.17.2.5 __float() . . . 72
5.17.2.6 __integer() . . . 72
5.17.2.7 __lessThan() . . . 72
5.17.2.8 __modulo() . . . 73
5.17.2.9 __multiply() . . . 73
5.17.2.10 __negative() . . . 74
5.17.2.11 __not() . . . 74
5.17.2.12 __string() . . . 74
5.17.2.13 __subtract() . . . 74
5.17.2.14 dump() . . . 75
5.17.2.15 is_equal() [1/6] . . . 75
5.17.2.16 is_equal() [2/6] . . . 76
5.17.2.17 is_equal() [3/6] . . . 76
5.17.2.18 is_equal() [4/6] . . . 76
5.17.2.19 is_equal() [5/6] . . . 77
5.17.2.20 is_equal() [6/6] . . . 77
5.17.2.21 makeCopy() . . . 77
5.18 Tang::ComputedExpressionBoolean Class Reference . . . 78
5.18.1 Detailed Description . . . 79
5.18.2 Constructor & Destructor Documentation . . . 80
5.18.2.1 ComputedExpressionBoolean() . . . 80
5.18.3 Member Function Documentation . . . 80
5.18.3.1 __add() . . . 80
5.18.3.2 __boolean() . . . 80
5.18.3.3 __divide() . . . 81
5.18.3.4 __equal() . . . 81
5.18.3.5 __float() . . . 81
5.18.3.6 __integer() . . . 82
5.18.3.7 __lessThan() . . . 82
5.18.3.8 __modulo() . . . 82
5.18.3.9 __multiply() . . . 83
5.18.3.10 __negative() . . . 83
5.18.3.11 __not() . . . 83
5.18.3.12 __string() . . . 84
5.18.3.13 __subtract() . . . 84
5.18.3.14 dump() . . . 84

5.18.3.15 is_equal() [1/6] . . . 84
5.18.3.16 is_equal() [2/6] . . . 85
5.18.3.17 is_equal() [3/6] . . . 85
5.18.3.18 is_equal() [4/6] . . . 86
5.18.3.19 is_equal() [5/6] . . . 86
5.18.3.20 is_equal() [6/6] . . . 86
5.18.3.21 makeCopy() . . . 87
5.19 Tang::ComputedExpressionError Class Reference . . . 87
5.19.1 Detailed Description . . . 89
5.19.2 Constructor & Destructor Documentation . . . 89
5.19.2.1 ComputedExpressionError() . . . 89
5.19.3 Member Function Documentation . . . 89
5.19.3.1 __add() . . . 89
5.19.3.2 __boolean() . . . 90
5.19.3.3 __divide() . . . 90
5.19.3.4 __equal() . . . 90
5.19.3.5 __float() . . . 91
5.19.3.6 __integer() . . . 91
5.19.3.7 __lessThan() . . . 91
5.19.3.8 __modulo() . . . 92
5.19.3.9 __multiply() . . . 92
5.19.3.10 __negative() . . . 93
5.19.3.11 __not() . . . 93
5.19.3.12 __string() . . . 93
5.19.3.13 __subtract() . . . 93
5.19.3.14 dump() . . . 94
5.19.3.15 is_equal() [1/6] . . . 94
5.19.3.16 is_equal() [2/6] . . . 95
5.19.3.17 is_equal() [3/6] . . . 96
5.19.3.18 is_equal() [4/6] . . . 96
5.19.3.19 is_equal() [5/6] . . . 97
5.19.3.20 is_equal() [6/6] . . . 97
5.19.3.21 makeCopy() . . . 97
5.20 Tang::ComputedExpressionFloat Class Reference . . . 98
5.20.1 Detailed Description . . . 99
5.20.2 Constructor & Destructor Documentation . . . 99
5.20.2.1 ComputedExpressionFloat() . . . 99
5.20.3 Member Function Documentation . . . 100
5.20.3.1 __add() . . . 100
5.20.3.2 __boolean() . . . 100
5.20.3.3 __divide() . . . 100
5.20.3.4 __equal() . . . 101

5.20.3.5 __float() . . . 101
5.20.3.6 __integer() . . . 102
5.20.3.7 __lessThan() . . . 102
5.20.3.8 __modulo() . . . 102
5.20.3.9 __multiply() . . . 103
5.20.3.10 __negative() . . . 103
5.20.3.11 __not() . . . 103
5.20.3.12 __string() . . . 104
5.20.3.13 __subtract() . . . 104
5.20.3.14 dump() . . . 104
5.20.3.15 is_equal() [1/6] . . . 105
5.20.3.16 is_equal() [2/6] . . . 105
5.20.3.17 is_equal() [3/6] . . . 105
5.20.3.18 is_equal() [4/6] . . . 106
5.20.3.19 is_equal() [5/6] . . . 106
5.20.3.20 is_equal() [6/6] . . . 107
5.20.3.21 makeCopy() . . . 107
5.21 Tang::ComputedExpressionInteger Class Reference . . . 107
5.21.1 Detailed Description . . . 109
5.21.2 Constructor & Destructor Documentation . . . 109
5.21.2.1 ComputedExpressionInteger() . . . 109
5.21.3 Member Function Documentation . . . 110
5.21.3.1 __add() . . . 110
5.21.3.2 __boolean() . . . 110
5.21.3.3 __divide() . . . 110
5.21.3.4 __equal() . . . 111
5.21.3.5 __float() . . . 111
5.21.3.6 __integer() . . . 112
5.21.3.7 __lessThan() . . . 112
5.21.3.8 __modulo() . . . 112
5.21.3.9 __multiply() . . . 113
5.21.3.10 __negative() . . . 113
5.21.3.11 __not() . . . 113
5.21.3.12 __string() . . . 114
5.21.3.13 __subtract() . . . 114
5.21.3.14 dump() . . . 114
5.21.3.15 is_equal() [1/6] . . . 115
5.21.3.16 is_equal() [2/6] . . . 115
5.21.3.17 is_equal() [3/6] . . . 115
5.21.3.18 is_equal() [4/6] . . . 116
5.21.3.19 is_equal() [5/6] . . . 116
5.21.3.20 is_equal() [6/6] . . . 117

5.21.3.21 makeCopy() . . . 117
5.22 Tang::ComputedExpressionString Class Reference . . . 117
5.22.1 Detailed Description . . . 119
5.22.2 Constructor & Destructor Documentation . . . 119
5.22.2.1 ComputedExpressionString() . . . 119
5.22.3 Member Function Documentation . . . 120
5.22.3.1 __add() . . . 120
5.22.3.2 __boolean() . . . 120
5.22.3.3 __divide() . . . 120
5.22.3.4 __equal() . . . 121
5.22.3.5 __float() . . . 121
5.22.3.6 __integer() . . . 121
5.22.3.7 __lessThan() . . . 122
5.22.3.8 __modulo() . . . 122
5.22.3.9 __multiply() . . . 122
5.22.3.10 __negative() . . . 123
5.22.3.11 __not() . . . 123
5.22.3.12 __string() . . . 123
5.22.3.13 __subtract() . . . 123
5.22.3.14 dump() . . . 124
5.22.3.15 is_equal() [1/6] . . . 124
5.22.3.16 is_equal() [2/6] . . . 124
5.22.3.17 is_equal() [3/6] . . . 125
5.22.3.18 is_equal() [4/6] . . . 125
5.22.3.19 is_equal() [5/6] . . . 126
5.22.3.20 is_equal() [6/6] . . . 126
5.22.3.21 makeCopy() . . . 126
5.23 Tang::Error Class Reference . . . 127
5.23.1 Detailed Description . . . 128
5.23.2 Constructor & Destructor Documentation . . . 128
5.23.2.1 Error() [1/2] . . . 128
5.23.2.2 Error() [2/2] . . . 128
5.23.3 Friends And Related Function Documentation . . . 128
5.23.3.1 operator<< . . . 129
5.24 Tang::GarbageCollected Class Reference . . . 129
5.24.1 Detailed Description . . . 131
5.24.2 Constructor & Destructor Documentation . . . 131
5.24.2.1 GarbageCollected() [1/3] . . . 131
5.24.2.2 GarbageCollected() [2/3] . . . 132
5.24.2.3 ~GarbageCollected() . . . 132
5.24.2.4 GarbageCollected() [3/3] . . . 132
5.24.3 Member Function Documentation . . . 132

5.24.3.1 make() . . . 132
5.24.3.2 operator"!() . . . 133
5.24.3.3 operator"!=() . . . 133
5.24.3.4 operator%() . . . 134
5.24.3.5 operator∗() [1/2] . . . 135
5.24.3.6 operator∗() [2/2] . . . 135
5.24.3.7 operator+() . . . 135
5.24.3.8 operator-() [1/2] . . . 136
5.24.3.9 operator-() [2/2] . . . 136
5.24.3.10 operator->() . . . 137
5.24.3.11 operator/() . . . 137
5.24.3.12 operator<() . . . 138
5.24.3.13 operator<=() . . . 138
5.24.3.14 operator=() [1/2] . . . 139
5.24.3.15 operator=() [2/2] . . . 139
5.24.3.16 operator==() [1/8] . . . 140
5.24.3.17 operator==() [2/8] . . . 140
5.24.3.18 operator==() [3/8] . . . 141
5.24.3.19 operator==() [4/8] . . . 141
5.24.3.20 operator==() [5/8] . . . 141
5.24.3.21 operator==() [6/8] . . . 142
5.24.3.22 operator==() [7/8] . . . 142
5.24.3.23 operator==() [8/8] . . . 143
5.24.3.24 operator>() . . . 143
5.24.3.25 operator>=() . . . 143
5.24.4 Friends And Related Function Documentation . . . 144
5.24.4.1 operator<< . . . 144
5.25 Tang::location Class Reference . . . 144
5.25.1 Detailed Description . . . 146
5.26 Tang::position Class Reference . . . 146
5.26.1 Detailed Description . . . 147
5.27 Tang::Program Class Reference . . . 147
5.27.1 Detailed Description . . . 149
5.27.2 Member Enumeration Documentation . . . 149
5.27.2.1 CodeType . . . 149
5.27.3 Constructor & Destructor Documentation . . . 149
5.27.3.1 Program() . . . 150
5.27.4 Member Function Documentation . . . 150
5.27.4.1 addBytecode() . . . 150
5.27.4.2 dumpBytecode() . . . 150
5.27.4.3 execute() . . . 151
5.27.4.4 getAst() . . . 151

5.27.4.5 getBytecode() . . . 151
5.27.4.6 getCode() . . . 151
5.27.4.7 getResult() . . . 152
5.27.4.8 setJumpTarget() . . . 152
5.28 Tang::SingletonObjectPool< T > Class Template Reference . . . 152
5.28.1 Detailed Description . . . 153
5.28.2 Member Function Documentation . . . 153
5.28.2.1 get() . . . 153
5.28.2.2 getInstance() . . . 153
5.28.2.3 recycle() . . . 153
5.29 Tang::TangBase Class Reference . . . 154
5.29.1 Detailed Description . . . 154
5.29.2 Constructor & Destructor Documentation . . . 154
5.29.2.1 TangBase() . . . 154
5.29.3 Member Function Documentation . . . 154
5.29.3.1 compileScript() . . . 154
5.30 Tang::TangScanner Class Reference . . . 155
5.30.1 Detailed Description . . . 156
5.30.2 Constructor & Destructor Documentation . . . 156
5.30.2.1 TangScanner() . . . 156
5.30.3 Member Function Documentation . . . 156
5.30.3.1 get_next_token() . . . 156

6 File Documentation **159**

6.1 build/generated/location.hh File Reference . . . 159
6.1.1 Detailed Description . . . 160
6.1.2 Function Documentation . . . 160
6.1.2.1 operator<<() [1/2] . . . 160
6.1.2.2 operator<<() [2/2] . . . 161
6.2 include/astNode.hpp File Reference . . . 161
6.2.1 Detailed Description . . . 162
6.3 include/astNodeAssign.hpp File Reference . . . 162
6.3.1 Detailed Description . . . 163
6.4 include/astNodeBinary.hpp File Reference . . . 163
6.4.1 Detailed Description . . . 164
6.5 include/astNodeBlock.hpp File Reference . . . 164
6.5.1 Detailed Description . . . 165
6.6 include/astNodeBoolean.hpp File Reference . . . 165
6.6.1 Detailed Description . . . 166
6.7 include/astNodeCast.hpp File Reference . . . 166
6.7.1 Detailed Description . . . 167
6.8 include/astNodeDoWhile.hpp File Reference . . . 167

6.8.1 Detailed Description . . . 168
6.9 include/astNodeFloat.hpp File Reference . . . 168
6.9.1 Detailed Description . . . 169
6.10 include/astNodeFor.hpp File Reference . . . 169
6.10.1 Detailed Description . . . 170
6.11 include/astNodeIdentifier.hpp File Reference . . . 170
6.11.1 Detailed Description . . . 171
6.12 include/astNodeIfElse.hpp File Reference . . . 171
6.12.1 Detailed Description . . . 172
6.13 include/astNodeInteger.hpp File Reference . . . 172
6.13.1 Detailed Description . . . 173
6.14 include/astNodePrint.hpp File Reference . . . 173
6.14.1 Detailed Description . . . 174
6.15 include/astNodeString.hpp File Reference . . . 174
6.15.1 Detailed Description . . . 175
6.16 include/astNodeUnary.hpp File Reference . . . 175
6.16.1 Detailed Description . . . 176
6.17 include/astNodeWhile.hpp File Reference . . . 176
6.17.1 Detailed Description . . . 177
6.18 include/computedExpression.hpp File Reference . . . 177
6.18.1 Detailed Description . . . 178
6.19 include/computedExpressionBoolean.hpp File Reference . . . 178
6.19.1 Detailed Description . . . 179
6.20 include/computedExpressionError.hpp File Reference . . . 179
6.20.1 Detailed Description . . . 180
6.21 include/computedExpressionFloat.hpp File Reference . . . 180
6.21.1 Detailed Description . . . 180
6.22 include/computedExpressionInteger.hpp File Reference . . . 181
6.22.1 Detailed Description . . . 181
6.23 include/computedExpressionString.hpp File Reference . . . 182
6.23.1 Detailed Description . . . 182
6.24 include/error.hpp File Reference . . . 183
6.24.1 Detailed Description . . . 183
6.25 include/garbageCollected.hpp File Reference . . . 184
6.25.1 Detailed Description . . . 184
6.26 include/macros.hpp File Reference . . . 184
6.26.1 Detailed Description . . . 185
6.26.2 Macro Definition Documentation . . . 185
6.26.2.1 TANG_UNUSED . . . 185
6.27 include/opcode.hpp File Reference . . . 185
6.27.1 Detailed Description . . . 186
6.27.2 Enumeration Type Documentation . . . 186

6.27.2.1 Opcode . . . 186
6.28 include/program.hpp File Reference . . . 187
6.28.1 Detailed Description . . . 187
6.29 include/singletonObjectPool.hpp File Reference . . . 188
6.29.1 Detailed Description . . . 188
6.30 include/tang.hpp File Reference . . . 189
6.30.1 Detailed Description . . . 189
6.31 include/tangBase.hpp File Reference . . . 190
6.31.1 Detailed Description . . . 191
6.32 include/tangScanner.hpp File Reference . . . 191
6.32.1 Detailed Description . . . 192
6.33 src/astNode.cpp File Reference . . . 192
6.33.1 Detailed Description . . . 192
6.34 src/astNodeAssign.cpp File Reference . . . 192
6.34.1 Detailed Description . . . 193
6.35 src/astNodeBinary.cpp File Reference . . . 193
6.35.1 Detailed Description . . . 194
6.36 src/astNodeBlock.cpp File Reference . . . 194
6.36.1 Detailed Description . . . 194
6.37 src/astNodeBoolean.cpp File Reference . . . 194
6.37.1 Detailed Description . . . 195
6.38 src/astNodeCast.cpp File Reference . . . 195
6.38.1 Detailed Description . . . 196
6.39 src/astNodeDoWhile.cpp File Reference . . . 196
6.39.1 Detailed Description . . . 197
6.40 src/astNodeFloat.cpp File Reference . . . 197
6.40.1 Detailed Description . . . 198
6.41 src/astNodeFor.cpp File Reference . . . 198
6.41.1 Detailed Description . . . 198
6.42 src/astNodeIdentifier.cpp File Reference . . . 198
6.42.1 Detailed Description . . . 199
6.43 src/astNodeIfElse.cpp File Reference . . . 199
6.43.1 Detailed Description . . . 200
6.44 src/astNodeInteger.cpp File Reference . . . 200
6.44.1 Detailed Description . . . 201
6.45 src/astNodePrint.cpp File Reference . . . 201
6.45.1 Detailed Description . . . 201
6.46 src/astNodeString.cpp File Reference . . . 201
6.46.1 Detailed Description . . . 202
6.47 src/astNodeUnary.cpp File Reference . . . 202
6.47.1 Detailed Description . . . 203
6.48 src/astNodeWhile.cpp File Reference . . . 203

6.48.1 Detailed Description . . . 204
6.49 src/computedExpression.cpp File Reference . . . 204
6.49.1 Detailed Description . . . 204
6.50 src/computedExpressionBoolean.cpp File Reference . . . 204
6.50.1 Detailed Description . . . 205
6.51 src/computedExpressionError.cpp File Reference . . . 205
6.51.1 Detailed Description . . . 206
6.52 src/computedExpressionFloat.cpp File Reference . . . 206
6.52.1 Detailed Description . . . 206
6.53 src/computedExpressionInteger.cpp File Reference . . . 206
6.53.1 Detailed Description . . . 207
6.54 src/computedExpressionString.cpp File Reference . . . 207
6.54.1 Detailed Description . . . 208
6.55 src/error.cpp File Reference . . . 208
6.55.1 Detailed Description . . . 208
6.55.2 Function Documentation . . . 208
6.55.2.1 operator<<() . . . 208
6.56 src/program-dumpBytecode.cpp File Reference . . . 209
6.56.1 Detailed Description . . . 209
6.56.2 Macro Definition Documentation . . . 210
6.56.2.1 DUMPPROGRAMCHECK . . . 210
6.57 src/program-execute.cpp File Reference . . . 210
6.57.1 Detailed Description . . . 211
6.57.2 Macro Definition Documentation . . . 211
6.57.2.1 EXECUTEPROGRAMCHECK . . . 211
6.57.2.2 STACKCHECK . . . 211
6.58 src/program.cpp File Reference . . . 212
6.58.1 Detailed Description . . . 212
6.59 src/tangBase.cpp File Reference . . . 212
6.59.1 Detailed Description . . . 213
6.60 test/test.cpp File Reference . . . 213
6.60.1 Detailed Description . . . 214
6.61 test/testGarbageCollected.cpp File Reference . . . 214
6.61.1 Detailed Description . . . 215
6.62 test/testSingletonObjectPool.cpp File Reference . . . 215
6.62.1 Detailed Description . . . 216

Index **217**

Chapter 1

Tang: A Template Language

1.1 Quick Description

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

- YouTube playlist
- GitHub repository

1.2 Features

The following features are planned:

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

1.3 License


```
MIT License

Copyright (c) 2022 Corey Pennycuff

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

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

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Tang::AstNode . . . 11
- Tang::AstNodeAssign . . . 15
- Tang::AstNodeBinary . . . 18
- Tang::AstNodeBlock . . . 22
- Tang::AstNodeBoolean . . . 26
- Tang::AstNodeCast . . . 30
- Tang::AstNodeDoWhile . . . 33
- Tang::AstNodeFloat . . . 37
- Tang::AstNodeFor . . . 40
- Tang::AstNodeIdentifier . . . 44
- Tang::AstNodeIfElse . . . 47
- Tang::AstNodeInteger . . . 51
- Tang::AstNodePrint . . . 54
- Tang::AstNodeString . . . 57
- Tang::AstNodeUnary . . . 61
- Tang::AstNodeWhile . . . 65

Tang::ComputedExpression . . . 69
- Tang::ComputedExpressionBoolean . . . 78
- Tang::ComputedExpressionError . . . 87
- Tang::ComputedExpressionFloat . . . 98
- Tang::ComputedExpressionInteger . . . 107
- Tang::ComputedExpressionString . . . 117

Tang::Error . . . 127
Tang::GarbageCollected . . . 129
Tang::location . . . 144
Tang::position . . . 146
Tang::Program . . . 147
Tang::SingletonObjectPool< T > . . . 152
Tang::TangBase . . . 154
TangTangFlexLexer
- Tang::TangScanner . . . 155

Chapter 3

Class Index

3.1 Class List

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

Tang::AstNode
: Base class for representing nodes of an Abstract Syntax Tree (AST) . . . 11

Tang::AstNodeAssign
: An AstNode that represents a binary expression . . . 15

Tang::AstNodeBinary
: An AstNode that represents a binary expression . . . 18

Tang::AstNodeBlock
: An AstNode that represents a code block . . . 22

Tang::AstNodeBoolean
: An AstNode that represents a boolean literal . . . 26

Tang::AstNodeCast
: An AstNode that represents a typecast of an expression . . . 30

Tang::AstNodeDoWhile
: An AstNode that represents a do..while statement . . . 33

Tang::AstNodeFloat
: An AstNode that represents an float literal . . . 37

Tang::AstNodeFor
: An AstNode that represents an if() statement . . . 40

Tang::AstNodeIdentifier
: An AstNode that represents an identifier . . . 44

Tang::AstNodeIfElse
: An AstNode that represents an if..else statement . . . 47

Tang::AstNodeInteger
: An AstNode that represents an integer literal . . . 51

Tang::AstNodePrint
: An AstNode that represents a print typeeration . . . 54

Tang::AstNodeString
: An AstNode that represents a string literal . . . 57

Tang::AstNodeUnary
: An AstNode that represents a unary negation . . . 61

Tang::AstNodeWhile
: An AstNode that represents a while statement . . . 65

Tang::ComputedExpression
: Represents the result of a computation that has been executed . . . 69

Tang::ComputedExpressionBoolean
: Represents an Boolean that is the result of a computation . . . 78

Tang::ComputedExpressionError
Represents a Runtime Error . . . 87
Tang::ComputedExpressionFloat
Represents a Float that is the result of a computation . . . 98
Tang::ComputedExpressionInteger
Represents an Integer that is the result of a computation . . . 107
Tang::ComputedExpressionString
Represents a String that is the result of a computation . . . 117
Tang::Error
Used to report any error of the system, whether a syntax (parsing) error or a runtime (execution) error . . . 127
Tang::GarbageCollected
A container that acts as a resource-counting garbage collector for the specified type . . . 129
Tang::location
Two points in a source file . . . 144
Tang::position
A point in a source file . . . 146
Tang::Program
Represents a compiled script or template that may be executed . . . 147
Tang::SingletonObjectPool< T >
A thread-safe, singleton object pool of the designated type . . . 152
Tang::TangBase
The base class for the Tang programming language . . . 154
Tang::TangScanner
The Flex lexer class for the main Tang language . . . 155

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 . . . 159
include/astNode.hpp
Declare the Tang::AstNode base class . . . 161
include/astNodeAssign.hpp
Declare the Tang::AstNodeAssign class . . . 162
include/astNodeBinary.hpp
Declare the Tang::AstNodeBinary class . . . 163
include/astNodeBlock.hpp
Declare the Tang::AstNodeBlock class . . . 164
include/astNodeBoolean.hpp
Declare the Tang::AstNodeBoolean class . . . 165
include/astNodeCast.hpp
Declare the Tang::AstNodeCast class . . . 166
include/astNodeDoWhile.hpp
Declare the Tang::AstNodeDoWhile class . . . 167
include/astNodeFloat.hpp
Declare the Tang::AstNodeFloat class . . . 168
include/astNodeFor.hpp
Declare the Tang::AstNodeFor class . . . 169
include/astNodeIdentifier.hpp
Declare the Tang::AstNodeIdentifier class . . . 170
include/astNodeIfElse.hpp
Declare the Tang::AstNodeIfElse class . . . 171
include/astNodeInteger.hpp
Declare the Tang::AstNodeInteger class . . . 172
include/astNodePrint.hpp
Declare the Tang::AstNodePrint class . . . 173
include/astNodeString.hpp
Declare the Tang::AstNodeString class . . . 174
include/astNodeUnary.hpp
Declare the Tang::AstNodeUnary class . . . 175
include/astNodeWhile.hpp
Declare the Tang::AstNodeWhile class . . . 176
include/computedExpression.hpp
Declare the Tang::ComputedExpression base class . . . 177

include/computedExpressionBoolean.hpp
Declare the Tang::ComputedExpressionBoolean class . . . 178
include/computedExpressionError.hpp
Declare the Tang::ComputedExpressionError class . . . 179
include/computedExpressionFloat.hpp
Declare the Tang::ComputedExpressionFloat class . . . 180
include/computedExpressionInteger.hpp
Declare the Tang::ComputedExpressionInteger class . . . 181
include/computedExpressionString.hpp
Declare the Tang::ComputedExpressionString class . . . 182
include/error.hpp
Declare the Tang::Error class used to describe syntax and runtime errors . . . 183
include/garbageCollected.hpp
Declare the Tang::GarbageCollected class . . . 184
include/macros.hpp
Contains generic macros . . . 184
include/opcode.hpp
Declare the Opcodes used in the Bytecode representation of a program . . . 185
include/program.hpp
Declare the Tang::Program class used to compile and execute source code . . . 187
include/singletonObjectPool.hpp
Declare the Tang::SingletonObjectPool class . . . 188
include/tang.hpp
Header file supplied for use by 3rd party code so that they can easily include all necessary headers . . . 189
include/tangBase.hpp
Declare the Tang::TangBase class used to interact with Tang . . . 190
include/tangScanner.hpp
Declare the Tang::TangScanner used to tokenize a Tang script . . . 191
src/astNode.cpp
Define the Tang::AstNode class . . . 192
src/astNodeAssign.cpp
Define the Tang::AstNodeAssign class . . . 192
src/astNodeBinary.cpp
Define the Tang::AstNodeBinary class . . . 193
src/astNodeBlock.cpp
Define the Tang::AstNodeBlock class . . . 194
src/astNodeBoolean.cpp
Define the Tang::AstNodeBoolean class . . . 194
src/astNodeCast.cpp
Define the Tang::AstNodeCast class . . . 195
src/astNodeDoWhile.cpp
Define the Tang::AstNodeDoWhile class . . . 196
src/astNodeFloat.cpp
Define the Tang::AstNodeFloat class . . . 197
src/astNodeFor.cpp
Define the Tang::AstNodeFor class . . . 198
src/astNodeIdentifier.cpp
Define the Tang::AstNodeIdentifier class . . . 198
src/astNodeIfElse.cpp
Define the Tang::AstNodeIfElse class . . . 199
src/astNodeInteger.cpp
Define the Tang::AstNodeInteger class . . . 200
src/astNodePrint.cpp
Define the Tang::AstNodePrint class . . . 201
src/astNodeString.cpp
Define the Tang::AstNodeString class . . . 201

src/astNodeUnary.cpp
Define the Tang::AstNodeUnary class . . . 202
src/astNodeWhile.cpp
Define the Tang::AstNodeWhile class . . . 203
src/computedExpression.cpp
Define the Tang::ComputedExpression class . . . 204
src/computedExpressionBoolean.cpp
Define the Tang::ComputedExpressionBoolean class . . . 204
src/computedExpressionError.cpp
Define the Tang::ComputedExpressionError class . . . 205
src/computedExpressionFloat.cpp
Define the Tang::ComputedExpressionFloat class . . . 206
src/computedExpressionInteger.cpp
Define the Tang::ComputedExpressionInteger class . . . 206
src/computedExpressionString.cpp
Define the Tang::ComputedExpressionString class . . . 207
src/error.cpp
Define the Tang::Error class . . . 208
src/program-dumpBytecode.cpp
Define the Tang::Program::dumpBytecode method . . . 209
src/program-execute.cpp
Define the Tang::Program::execute method . . . 210
src/program.cpp
Define the Tang::Program class . . . 212
src/tangBase.cpp
Define the Tang::TangBase class . . . 212
test/test.cpp
Test the general language behaviors . . . 213
test/testGarbageCollected.cpp
Test the generic behavior of the Tang::GarbageCollected class . . . 214
test/testSingletonObjectPool.cpp
Test the generic behavior of the Tang::SingletonObjectPool class . . . 215

Chapter 5

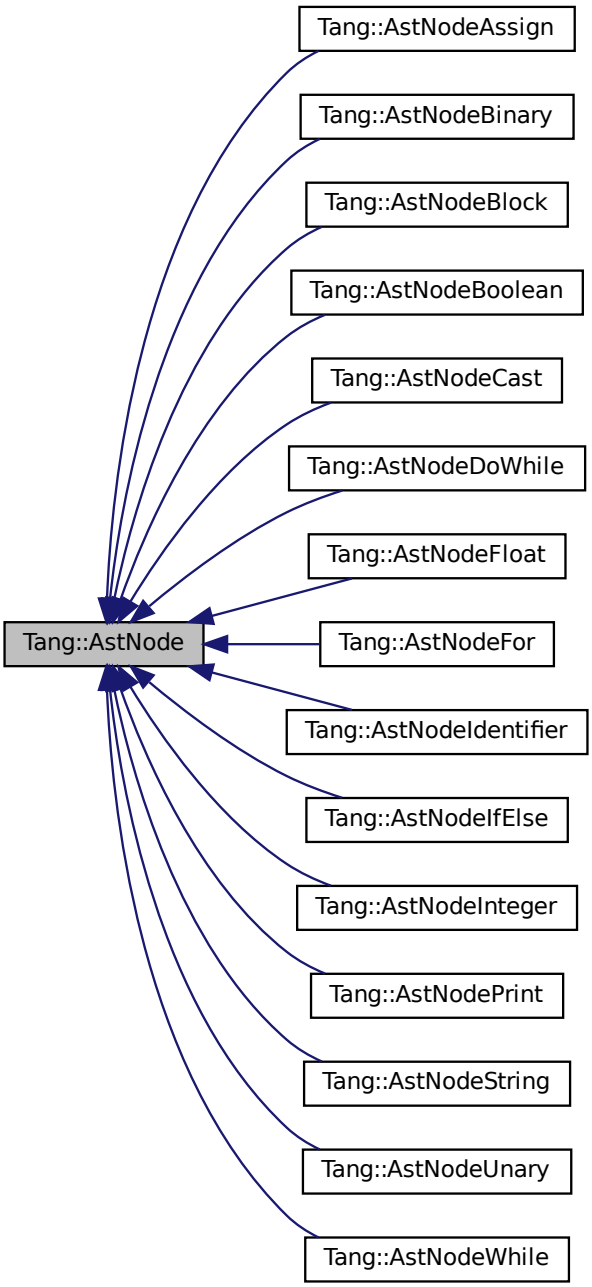
Class Documentation

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:



Public Member Functions

- AstNode (Tang::location location)

  The generic constructor.
- 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 collectIdentifiers (Program &program) const

  Compile a list of all variables in the scope.

- virtual void collectStrings (Program &program) const

  Compile a list of all string constants in the scope.

5.1.1 Detailed Description

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

By default, it will represent a NULL value. 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 )
```

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 collectIdentifiers()

```
void AstNode::collectIdentifiers (
            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::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.1.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.1.3.3 compile()

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

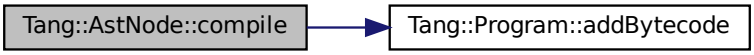
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeInteger, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeFloat, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBoolean, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

Here is the call graph for this function:



5.1.3.4 dump()

```
string AstNode::dump (
            std::string indent = "" ) const  [virtual]
```

Return a string that describes the contents of the node.

Parameters

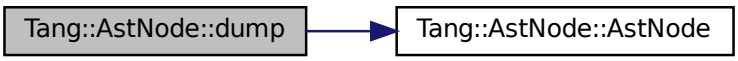
indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeInteger, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeFloat, Tang::AstNodeDoWhile, Tang::AstNodeCast, Tang::AstNodeBoolean, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

Here is the call graph for this function:



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

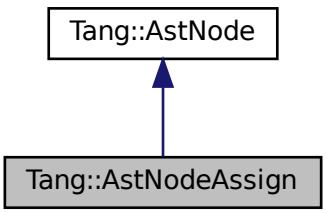
- include/astNode.hpp
- src/astNode.cpp

5.2 Tang::AstNodeAssign Class Reference

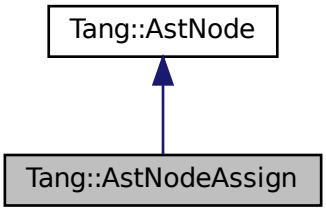
An AstNode that represents a binary expression.

`#include <astNodeAssign.hpp>`

Inheritance diagram for Tang::AstNodeAssign:



Collaboration diagram for Tang::AstNodeAssign:



Public Member Functions

- AstNodeAssign (std::shared_ptr< AstNode > lhs, std::shared_ptr< AstNode > rhs, Tang::location location)
  The constructor.
- virtual std::string dump (std::string indent="") const override
  Return a string that describes the contents of the node.
- virtual void compile (Tang::Program &program) const override
  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const override
  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override
  Compile a list of all string constants in the scope.

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 collectIdentifiers()

```
void AstNodeAssign::collectIdentifiers (
            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.

5.2.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.2.3.3 compile()

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

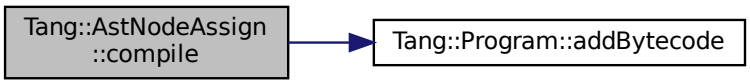
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.2.3.4 dump()

```
string AstNodeAssign::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

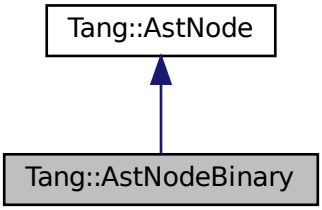
- include/astNodeAssign.hpp
- src/astNodeAssign.cpp

5.3 Tang::AstNodeBinary Class Reference

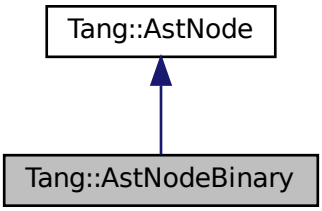
An AstNode that represents a binary expression.

```
#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 , And ,
  Or }

  Indicates the type of binary expression that this node represents.

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 collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

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
```

Indicates the type of binary expression that this node represents.

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.
And	Indicates lhs && rhs with short-circuit evaluation.
Or	Indicates lhs || rhs with short-circuit evaluation.

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 collectIdentifiers()

```
void AstNodeBinary::collectIdentifiers (
            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.

5.3.4.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.3.4.3 compile()

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

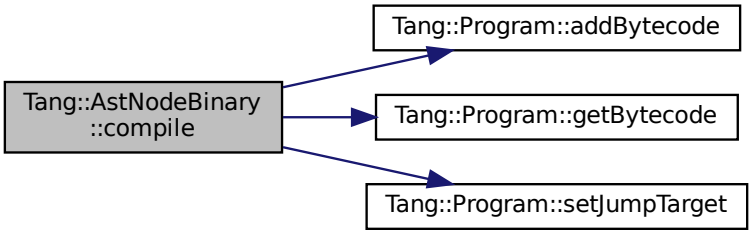
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.3.4.4 dump()

```
string AstNodeBinary::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

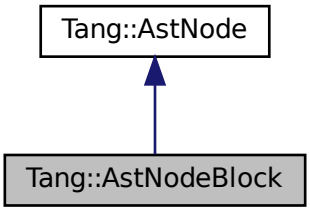
- include/astNodeBinary.hpp
- src/astNodeBinary.cpp

5.4 Tang::AstNodeBlock Class Reference

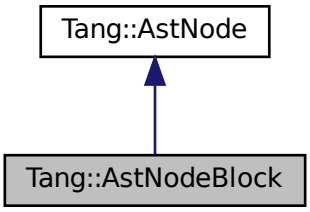
An AstNode that represents a code block.

```
#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 collectIdentifiers (Program &program) const override
  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override
  Compile a list of all string constants in the scope.

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 collectIdentifiers()

```
void AstNodeBlock::collectIdentifiers (
            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.

5.4.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.4.3.3 compile()

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

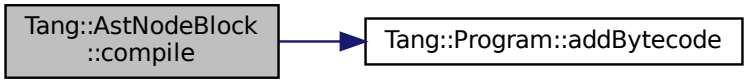
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.4.3.4 dump()

```
string AstNodeBlock::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

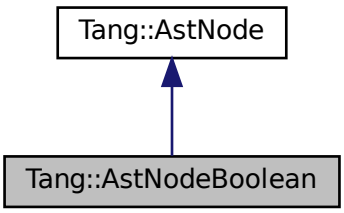
- include/astNodeBlock.hpp
- src/astNodeBlock.cpp

5.5 Tang::AstNodeBoolean Class Reference

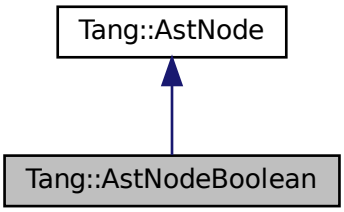
An AstNode that represents a boolean literal.

`#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 collectIdentifiers (Program &program) const
  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const
  Compile a list of all string constants in the scope.

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 collectIdentifiers()

```
void AstNode::collectIdentifiers (
            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::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.5.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.5.3.3 compile()

```
void AstNodeBoolean::compile (
            Tang::Program & program ) const  [override], [virtual]
```

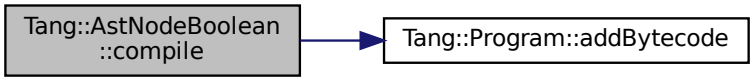
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.5.3.4 dump()

```
string AstNodeBoolean::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

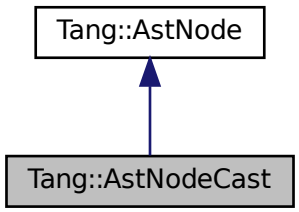
- include/astNodeBoolean.hpp
- src/astNodeBoolean.cpp

5.6 Tang::AstNodeCast Class Reference

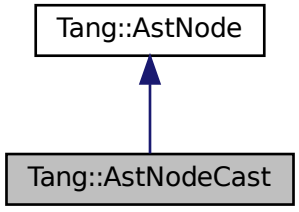
An AstNode that represents a typecast of an expression.

`#include <astNodeCast.hpp>`

Inheritance diagram for Tang::AstNodeCast:



Collaboration diagram for Tang::AstNodeCast:



Public Types

- enum Type { Integer , Float , Boolean }

  The possible types that can be cast to.

Public Member Functions

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

  The constructor.
- virtual std::string dump (std::string indent="") const override

  Return a string that describes the contents of the node.
- virtual void compile (Tang::Program &program) const override

  Compile the ast of the provided Tang::Program.
- virtual void collectIdentifiers (Program &program) const

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const

  Compile a list of all string constants in the scope.

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 collectIdentifiers()

```
void AstNode::collectIdentifiers (
            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::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.6.4.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.6.4.3 compile()

```
void AstNodeCast::compile (
            Tang::Program & program ) const  [override], [virtual]
```

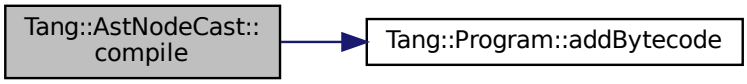
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.6.4.4 dump()

```
string AstNodeCast::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

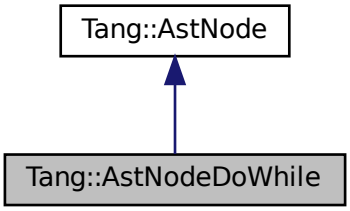
- include/astNodeCast.hpp
- src/astNodeCast.cpp

5.7 Tang::AstNodeDoWhile Class Reference

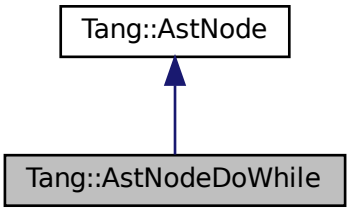
An AstNode that represents a do..while statement.

`#include <astNodeDoWhile.hpp>`

Inheritance diagram for Tang::AstNodeDoWhile:



Collaboration diagram for Tang::AstNodeDoWhile:



Public Member Functions

- AstNodeDoWhile (shared_ptr< AstNode > condition, shared_ptr< AstNode > codeBlock, 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 collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

5.7.1 Detailed Description

An AstNode that represents a do..while statement.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 AstNodeDoWhile()

```
AstNodeDoWhile::AstNodeDoWhile (
            shared_ptr< AstNode > condition,
            shared_ptr< AstNode > codeBlock,
            Tang::location location )
```

The constructor.

Parameters

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

5.7.3 Member Function Documentation

5.7.3.1 collectIdentifiers()

```
void AstNodeDoWhile::collectIdentifiers (
            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.

5.7.3.2 collectStrings()

```
void AstNodeDoWhile::collectStrings (
            Program & program ) const  [override], [virtual]
```

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.7.3.3 compile()

```
void AstNodeDoWhile::compile (
            Tang::Program & program ) const  [override], [virtual]
```

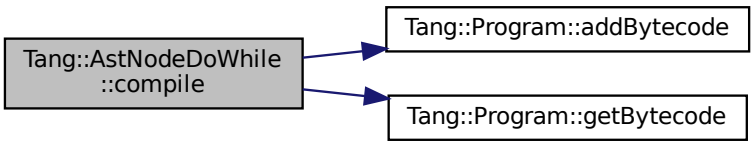
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.7.3.4 dump()

```
string AstNodeDoWhile::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

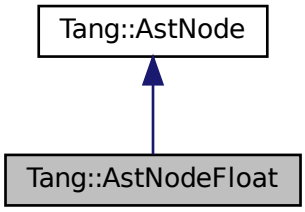
- include/astNodeDoWhile.hpp
- src/astNodeDoWhile.cpp

5.8 Tang::AstNodeFloat Class Reference

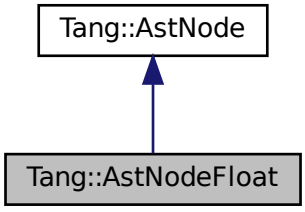
An AstNode that represents an float literal.

`#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 collectIdentifiers (Program &program) const

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const

  Compile a list of all string constants in the scope.

5.8.1 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.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 AstNodeFloat()

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

The constructor.

Parameters

number	The number to represent.
location	The location associated with the expression.

5.8.3 Member Function Documentation

5.8.3.1 collectIdentifiers()

```
void AstNode::collectIdentifiers (
            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::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.8.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.8.3.3 compile()

```
void AstNodeFloat::compile (
            Tang::Program & program ) const  [override], [virtual]
```

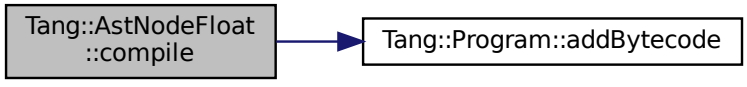
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.8.3.4 dump()

```
string AstNodeFloat::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

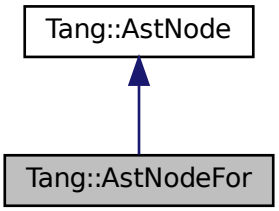
- include/astNodeFloat.hpp
- src/astNodeFloat.cpp

5.9 Tang::AstNodeFor Class Reference

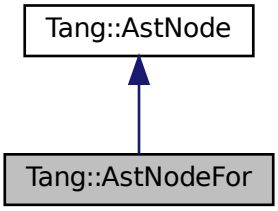
An AstNode that represents an if() statement.

```
#include <astNodeFor.hpp>
```

Inheritance diagram for Tang::AstNodeFor:



Collaboration diagram for Tang::AstNodeFor:



Public Member Functions

- AstNodeFor (shared_ptr< AstNode > initialization, shared_ptr< AstNode > condition, shared_ptr< AstNode > increment, shared_ptr< AstNode > codeBlock, 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 collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

5.9.1 Detailed Description

An AstNode that represents an if() statement.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 AstNodeFor()

```
AstNodeFor::AstNodeFor (
            shared_ptr< AstNode > initialization,
            shared_ptr< AstNode > condition,
            shared_ptr< AstNode > increment,
            shared_ptr< AstNode > codeBlock,
            Tang::location location )
```

The constructor.

Parameters

initialization	The expression to be executed first.
condition	The expression which determines whether the codeBlock is executed.
increment	The expression to be executed after each codeBlock.
codeBlock	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 collectIdentifiers()

```
void AstNodeFor::collectIdentifiers (
            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.

5.9.3.2 collectStrings()

```
void AstNodeFor::collectStrings (
            Program & program ) const  [override], [virtual]
```

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.9.3.3 compile()

```
void AstNodeFor::compile (
            Tang::Program & program ) const  [override], [virtual]
```

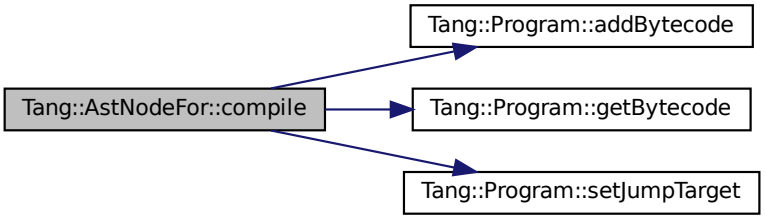
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.9.3.4 dump()

```
string AstNodeFor::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

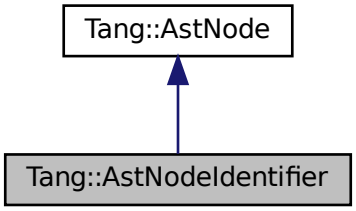
- include/astNodeFor.hpp
- src/astNodeFor.cpp

5.10 Tang::AstNodeIdentifier Class Reference

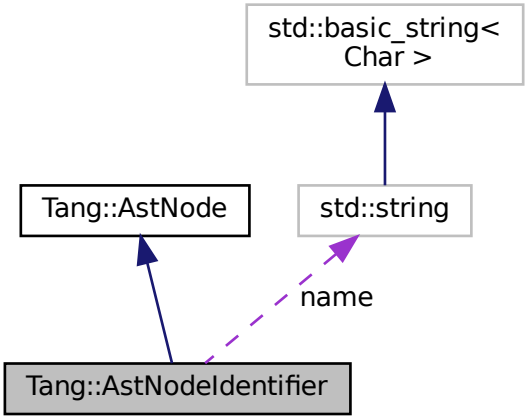
An AstNode that represents an identifier.

`#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 collectIdentifiers (Program &program) const override
  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const
  Compile a list of all string constants in the scope.

Public Attributes

- std::string name

  The name of the identifier.

5.10.1 Detailed Description

An AstNode that represents an identifier.

Identifier names are represented by a string.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 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.

5.10.3 Member Function Documentation

5.10.3.1 collectIdentifiers()

```
void AstNodeIdentifier::collectIdentifiers (
            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.

5.10.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.10.3.3 compile()

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

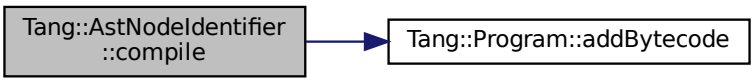
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.10.3.4 dump()

```
string AstNodeIdentifier::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

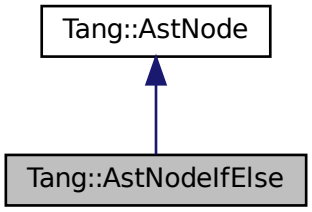
- include/astNodeIdentifier.hpp
- src/astNodeIdentifier.cpp

5.11 Tang::AstNodeIfElse Class Reference

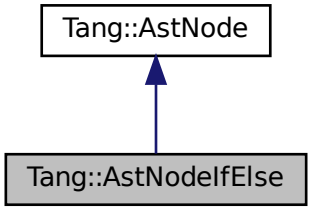
An AstNode that represents an if..else statement.

`#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 collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

5.11.1 Detailed Description

An AstNode that represents an if..else statement.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 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.

5.11.2.2 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.

5.11.3 Member Function Documentation

5.11.3.1 collectIdentifiers()

```
void AstNodeIfElse::collectIdentifiers (
            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.

5.11.3.2 collectStrings()

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

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.11.3.3 compile()

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

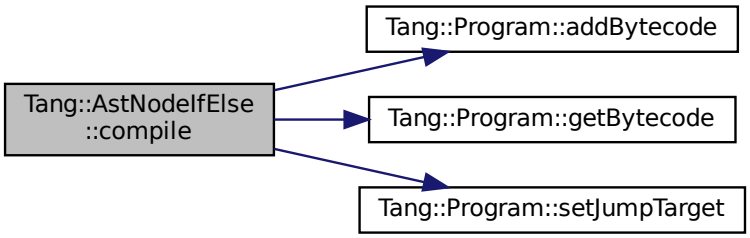
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.11.3.4 dump()

```
string AstNodeIfElse::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

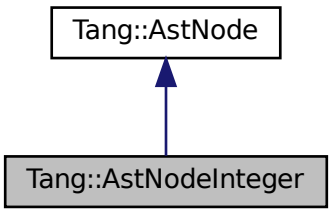
- include/astNodeIfElse.hpp
- src/astNodeIfElse.cpp

5.12 Tang::AstNodeInteger Class Reference

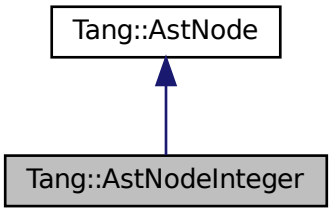
An AstNode that represents an integer literal.

`#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 collectIdentifiers (Program &program) const
  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const
  Compile a list of all string constants in the scope.

5.12.1 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.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 AstNodeInteger()

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

The constructor.

Parameters

number	The number to represent.
location	The location associated with the expression.

5.12.3 Member Function Documentation

5.12.3.1 collectIdentifiers()

```
void AstNode::collectIdentifiers (
            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::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.12.3.2 collectStrings()

```
void AstNode::collectStrings (
```

```
            Program & program ) const  [virtual], [inherited]
```

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented in Tang::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodeString, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.12.3.3 compile()

```
void AstNodeInteger::compile (
            Tang::Program & program ) const  [override], [virtual]
```

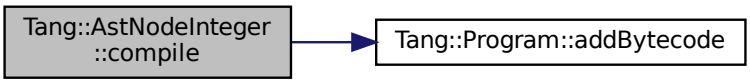
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.12.3.4 dump()

```
string AstNodeInteger::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

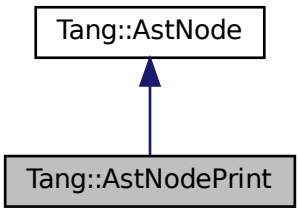
- include/astNodeInteger.hpp
- src/astNodeInteger.cpp

5.13 Tang::AstNodePrint Class Reference

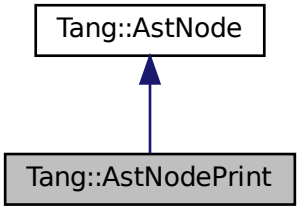
An AstNode that represents a print typeeration.

`#include <astNodePrint.hpp>`

Inheritance diagram for Tang::AstNodePrint:



Collaboration diagram for Tang::AstNodePrint:



Public Types

- enum Type { Default }

  The type of print() requested.

Public Member Functions

- AstNodePrint (Type type, 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 collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

5.13.1 Detailed Description

An AstNode that represents a print typeeration.

5.13.2 Member Enumeration Documentation

5.13.2.1 Type

```
enum Tang::AstNodePrint::Type
```

The type of print() requested.

Enumerator

Default	Use the default print.

5.13.3 Constructor & Destructor Documentation

5.13.3.1 AstNodePrint()

```
AstNodePrint::AstNodePrint (
            Type type,
            shared_ptr< AstNode > expression,
            Tang::location location )
```

The constructor.

Parameters

type	The Tang::AstNodePrint::Type being requested.
expression	The expression to be printed.
location	The location associated with the expression.

5.13.4 Member Function Documentation

5.13.4.1 collectIdentifiers()

```
void AstNodePrint::collectIdentifiers (
            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.

5.13.4.2 collectStrings()

```
void AstNodePrint::collectStrings (
            Program & program ) const  [override], [virtual]
```

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.13.4.3 compile()

```
void AstNodePrint::compile (
            Tang::Program & program ) const  [override], [virtual]
```

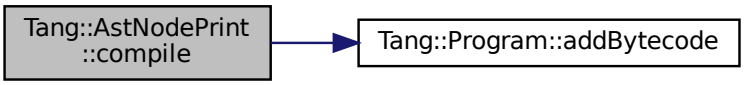
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.13.4.4 dump()

```
string AstNodePrint::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

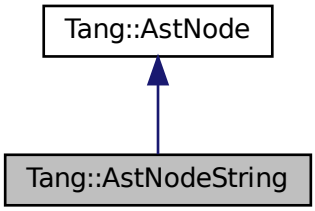
- include/astNodePrint.hpp
- src/astNodePrint.cpp

5.14 Tang::AstNodeString Class Reference

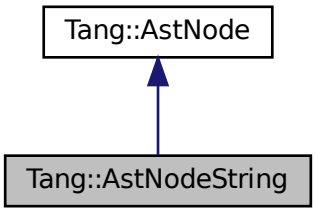
An AstNode that represents a string literal.

```
#include <astNodeString.hpp>
```

Inheritance diagram for Tang::AstNodeString:



Collaboration diagram for Tang::AstNodeString:



Public Member Functions

- AstNodeString (const string &text, 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 collectStrings (Program &program) const override
  Compile a list of all string constants in the scope.
- void compileLiteral (Tang::Program &program) const
  Compile the string and push it onto the stack.
- virtual void collectIdentifiers (Program &program) const
  Compile a list of all variables in the scope.

5.14.1 Detailed Description

An AstNode that represents a string literal.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 AstNodeString()

```
AstNodeString::AstNodeString (
            const string & text,
            Tang::location location )
```

The constructor.

Parameters

text	The string to represent.
location	The location associated with the expression.

5.14.3 Member Function Documentation

5.14.3.1 collectIdentifiers()

```
void AstNode::collectIdentifiers (
            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::AstNodeWhile, Tang::AstNodeUnary, Tang::AstNodePrint, Tang::AstNodeIfElse, Tang::AstNodeIdentifier, Tang::AstNodeFor, Tang::AstNodeDoWhile, Tang::AstNodeBlock, Tang::AstNodeBinary, and Tang::AstNodeAssign.

5.14.3.2 collectStrings()

```
void AstNodeString::collectStrings (
            Program & program ) const  [override], [virtual]
```

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.14.3.3 compile()

```
void AstNodeString::compile (
            Tang::Program & program ) const  [override], [virtual]
```

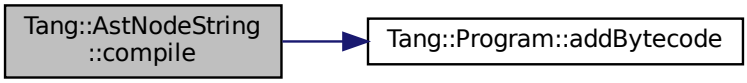
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.14.3.4 compileLiteral()

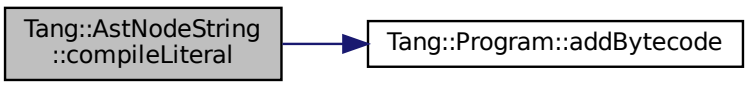
```
void AstNodeString::compileLiteral (
            Tang::Program & program ) const
```

Compile the string and push it onto the stack.

Parameters

program	The Program which will hold the generated Bytecode.

Here is the call graph for this function:



5.14.3.5 dump()

```
string AstNodeString::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

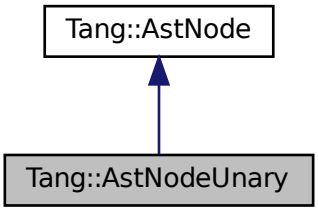
- include/astNodeString.hpp
- src/astNodeString.cpp

5.15 Tang::AstNodeUnary Class Reference

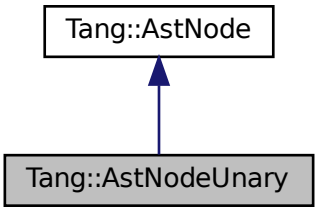
An AstNode that represents a unary negation.

```
#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 collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

5.15.1 Detailed Description

An AstNode that represents a unary negation.

5.15.2 Member Enumeration Documentation

5.15.2.1 Operator

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

The type of operation.

Enumerator

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

5.15.3 Constructor & Destructor Documentation

5.15.3.1 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.

5.15.4 Member Function Documentation

5.15.4.1 collectIdentifiers()

```
void AstNodeUnary::collectIdentifiers (
            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.

5.15.4.2 collectStrings()

```
void AstNodeUnary::collectStrings (
            Program & program ) const  [override], [virtual]
```

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.15.4.3 compile()

```
void AstNodeUnary::compile (
            Tang::Program & program ) const  [override], [virtual]
```

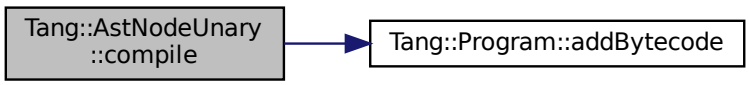
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.15.4.4 dump()

```
string AstNodeUnary::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

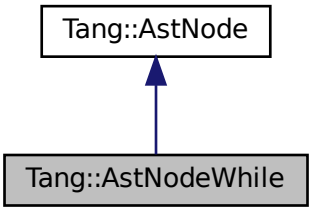
- include/astNodeUnary.hpp
- src/astNodeUnary.cpp

5.16 Tang::AstNodeWhile Class Reference

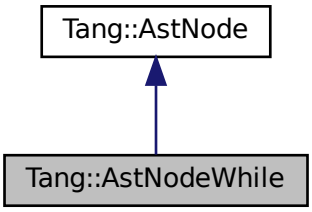
An AstNode that represents a while statement.

```
#include <astNodeWhile.hpp>
```

Inheritance diagram for Tang::AstNodeWhile:



Collaboration diagram for Tang::AstNodeWhile:



Public Member Functions

- AstNodeWhile (shared_ptr< AstNode > condition, shared_ptr< AstNode > codeBlock, 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 collectIdentifiers (Program &program) const override

  Compile a list of all variables in the scope.
- virtual void collectStrings (Program &program) const override

  Compile a list of all string constants in the scope.

5.16.1 Detailed Description

An AstNode that represents a while statement.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 AstNodeWhile()

```
AstNodeWhile::AstNodeWhile (
            shared_ptr< AstNode > condition,
            shared_ptr< AstNode > codeBlock,
            Tang::location location )
```

The constructor.

Parameters

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

5.16.3 Member Function Documentation

5.16.3.1 collectIdentifiers()

```
void AstNodeWhile::collectIdentifiers (
            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.

5.16.3.2 collectStrings()

```
void AstNodeWhile::collectStrings (
            Program & program ) const  [override], [virtual]
```

Compile a list of all string constants in the scope.

Parameters

program	The Tang::Program that is being compiled.

Reimplemented from Tang::AstNode.

5.16.3.3 compile()

```
void AstNodeWhile::compile (
            Tang::Program & program ) const  [override], [virtual]
```

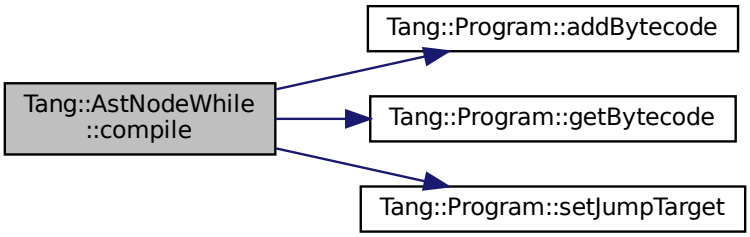
Compile the ast of the provided Tang::Program.

Parameters

program	The Program which will hold the generated Bytecode.

Reimplemented from Tang::AstNode.

Here is the call graph for this function:



5.16.3.4 dump()

```
string AstNodeWhile::dump (
            std::string indent = "" ) const  [override], [virtual]
```

Return a string that describes the contents of the node.

Parameters

indent	A string used to indent the dump.

Returns

The value as a string.

Reimplemented from Tang::AstNode.

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

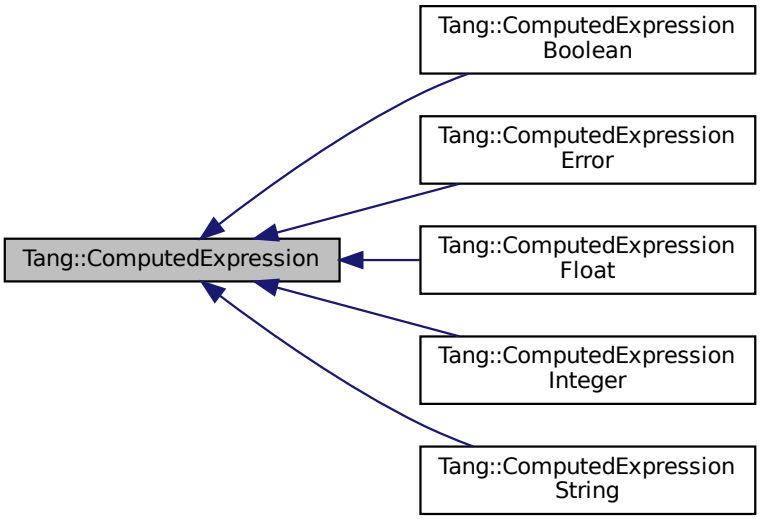
- include/astNodeWhile.hpp
- src/astNodeWhile.cpp

5.17 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 string &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.
- virtual GarbageCollected __string () const
  Perform a type cast to string.

5.17.1 Detailed Description

Represents the result of a computation that has been executed.

By default, it will represent a NULL value.

5.17.2 Member Function Documentation

5.17.2.1 __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::ComputedExpressionString, Tang::ComputedExpressionInteger, Tang::ComputedExpressionFloat, and Tang::ComputedExpressionError.

5.17.2.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

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

5.17.2.3 __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.

5.17.2.4 __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::ComputedExpressionString, Tang::ComputedExpressionInteger, Tang::ComputedExpressionFloat, Tang::ComputedExpressionError, and Tang::ComputedExpressionBoolean.

5.17.2.5 __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.

5.17.2.6 __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.

5.17.2.7 __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::ComputedExpressionString, Tang::ComputedExpressionInteger, Tang::ComputedExpressionFloat, and Tang::ComputedExpressionError.

5.17.2.8 __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.

5.17.2.9 __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.

5.17.2.10 __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.

5.17.2.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

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

5.17.2.12 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

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

5.17.2.13 __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.

5.17.2.14 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::ComputedExpressionString, Tang::ComputedExpressionInteger, Tang::ComputedExpressionFloat, Tang::ComputedExpressionError, and Tang::ComputedExpressionBoolean.

5.17.2.15 is_equal() [1/6]

```
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::ComputedExpressionString, Tang::ComputedExpressionInteger, Tang::ComputedExpressionFloat, and Tang::ComputedExpressionBoolean.

5.17.2.16 is_equal() [2/6]

```
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.

5.17.2.17 is_equal() [3/6]

```
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.

5.17.2.18 is_equal() [4/6]

```
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.

5.17.2.19 is_equal() [5/6]

```
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.

5.17.2.20 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const string & 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::ComputedExpressionString.

5.17.2.21 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::ComputedExpressionString, Tang::ComputedExpressionInteger, Tang::ComputedExpressionFloat, Tang::ComputedExpressionError, and Tang::ComputedExpressionBoolean.

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

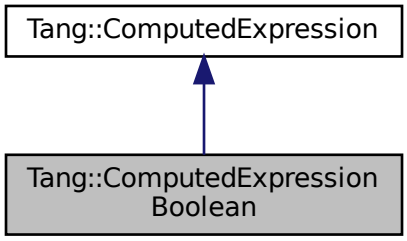
- include/computedExpression.hpp
- src/computedExpression.cpp

5.18 Tang::ComputedExpressionBoolean Class Reference

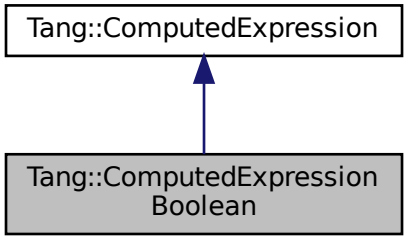
Represents an Boolean that is the result of a computation.

`#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 string &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.
- virtual GarbageCollected __string () const

  Perform a type cast to string.

5.18.1 Detailed Description

Represents an Boolean that is the result of a computation.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 ComputedExpressionBoolean()

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

Construct an Boolean result.

Parameters

val	The boolean value.

5.18.3 Member Function Documentation

5.18.3.1 __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::ComputedExpressionString, Tang::ComputedExpressionInteger, Tang::ComputedExpressionFloat, and Tang::ComputedExpressionError.

5.18.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.18.3.3 __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.

5.18.3.4 __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.

5.18.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.18.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.18.3.7 __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::ComputedExpressionString, Tang::ComputedExpressionInteger, Tang::ComputedExpressionFloat, and Tang::ComputedExpressionError.

5.18.3.8 __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.

5.18.3.9 __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.

5.18.3.10 __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.

5.18.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.18.3.12 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

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

5.18.3.13 __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.

5.18.3.14 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.

5.18.3.15 is_equal() [1/6]

```
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.

5.18.3.16 is_equal() [2/6]

```
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.

5.18.3.17 is_equal() [3/6]

```
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.

5.18.3.18 is_equal() [4/6]

```
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.

5.18.3.19 is_equal() [5/6]

```
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.

5.18.3.20 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const string & 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::ComputedExpressionString.

5.18.3.21 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:

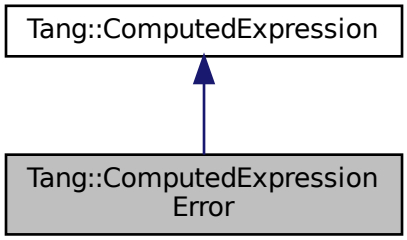
- include/computedExpressionBoolean.hpp
- src/computedExpressionBoolean.cpp

5.19 Tang::ComputedExpressionError Class Reference

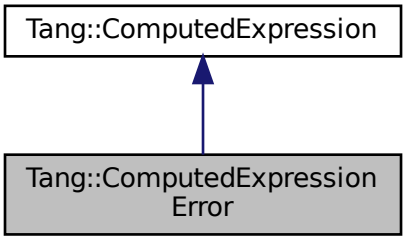
Represents a Runtime Error.

```
#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 GarbageCollected __string () const override

  Perform a type cast to string.
- 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 string &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.

5.19.1 Detailed Description

Represents a Runtime Error.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 ComputedExpressionError()

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

Construct a Runtime Error.

Parameters

error	The Tang::Error object.

5.19.3 Member Function Documentation

5.19.3.1 __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.

5.19.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.19.3.3 __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.

5.19.3.4 __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.

5.19.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.19.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.19.3.7 __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.

5.19.3.8 __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.

5.19.3.9 __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.

5.19.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.19.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.19.3.12 __string()

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

Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.19.3.13 __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.

5.19.3.14 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.

5.19.3.15 is_equal() [1/6]

```
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::ComputedExpressionString, Tang::ComputedExpressionInteger, Tang::ComputedExpressionFloat, and Tang::ComputedExpressionBoolean.

5.19.3.16 is_equal() [2/6]

```
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.

5.19.3.17 is_equal() [3/6]

```
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.

5.19.3.18 is_equal() [4/6]

```
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.

5.19.3.19 is_equal() [5/6]

```
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.

5.19.3.20 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const string & 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::ComputedExpressionString.

5.19.3.21 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:

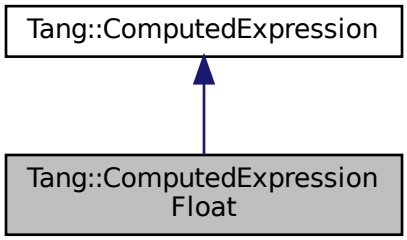
- include/computedExpressionError.hpp
- src/computedExpressionError.cpp

5.20 Tang::ComputedExpressionFloat Class Reference

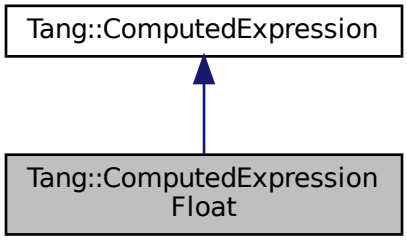
Represents a Float that is the result of a computation.

`#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 bool is_equal (const bool &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 GarbageCollected __string () const override

  Perform a type cast to string.
- virtual bool is_equal (const string &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**

5.20.1 Detailed Description

Represents a Float that is the result of a computation.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 ComputedExpressionFloat()

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

Construct a Float result.

Parameters

val	The float value.

5.20.3 Member Function Documentation

5.20.3.1 __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.

5.20.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.3 __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.

5.20.3.4 __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.

5.20.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.7 __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.

5.20.3.8 __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.

5.20.3.9 __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.

5.20.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.20.3.12 __string()

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

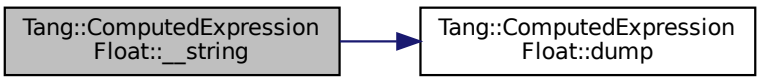
Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.20.3.13 __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.

5.20.3.14 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.

5.20.3.15 is_equal() [1/6]

```
bool ComputedExpressionFloat::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.

5.20.3.16 is_equal() [2/6]

```
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.

5.20.3.17 is_equal() [3/6]

```
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.

5.20.3.18 is_equal() [4/6]

```
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.

5.20.3.19 is_equal() [5/6]

```
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.

5.20.3.20 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const string & 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::ComputedExpressionString.

5.20.3.21 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:

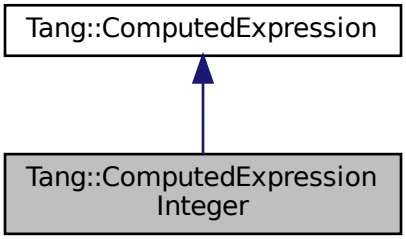
- include/computedExpressionFloat.hpp
- src/computedExpressionFloat.cpp

5.21 Tang::ComputedExpressionInteger Class Reference

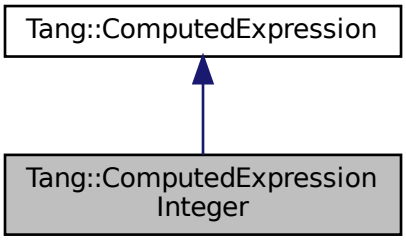
Represents an Integer that is the result of a computation.

```
#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 bool is_equal (const bool &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 GarbageCollected __string () const override

  Perform a type cast to string.
- virtual bool is_equal (const string &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**

5.21.1 Detailed Description

Represents an Integer that is the result of a computation.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 ComputedExpressionInteger()

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

Construct an Integer result.

Parameters

val	The integer value.

5.21.3 Member Function Documentation

5.21.3.1 __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.

5.21.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.3 __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.

5.21.3.4 __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.

5.21.3.5 __float()

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

Perform a type cast to float.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.6 __integer()

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

Perform a type cast to integer.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.7 __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.

5.21.3.8 __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.

5.21.3.9 __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.

5.21.3.10 __negative()

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

Compute the result of negating this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.21.3.12 __string()

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

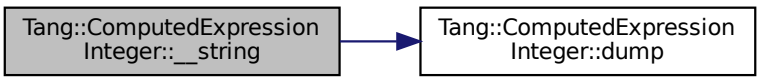
Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

Here is the call graph for this function:



5.21.3.13 __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.

5.21.3.14 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.

5.21.3.15 is_equal() [1/6]

```
bool ComputedExpressionInteger::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.

5.21.3.16 is_equal() [2/6]

```
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.

5.21.3.17 is_equal() [3/6]

```
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.

5.21.3.18 is_equal() [4/6]

```
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.

5.21.3.19 is_equal() [5/6]

```
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.

5.21.3.20 is_equal() [6/6]

```
virtual bool Tang::ComputedExpression::is_equal (
            const string & 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::ComputedExpressionString.

5.21.3.21 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:

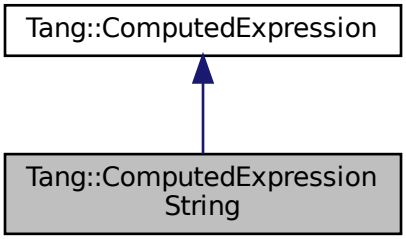
- include/computedExpressionInteger.hpp
- src/computedExpressionInteger.cpp

5.22 Tang::ComputedExpressionString Class Reference

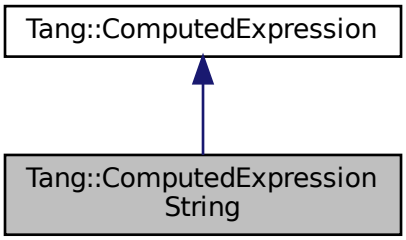
Represents a String that is the result of a computation.

```
#include <computedExpressionString.hpp>
```

Inheritance diagram for Tang::ComputedExpressionString:



Collaboration diagram for Tang::ComputedExpressionString:



Public Member Functions

- ComputedExpressionString (std::string val)

  Construct a String 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 bool is_equal (const string &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 __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 __boolean () const override

  Perform a type cast to boolean.
- virtual GarbageCollected __string () const override

  Perform a type cast to string.
- 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 __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 __integer () const

  Perform a type cast to integer.
- virtual GarbageCollected __float () const

  Perform a type cast to float.

5.22.1 Detailed Description

Represents a String that is the result of a computation.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 ComputedExpressionString()

```
ComputedExpressionString::ComputedExpressionString (
            std::string val )
```

Construct a String result.

Parameters

val	The string value.

5.22.3 Member Function Documentation

5.22.3.1 __add()

```
GarbageCollected ComputedExpressionString::__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.

5.22.3.2 __boolean()

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

Perform a type cast to boolean.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.3 __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.

5.22.3.4 __equal()

```
GarbageCollected ComputedExpressionString::__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.

5.22.3.5 __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.

5.22.3.6 __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.

5.22.3.7 __lessThan()

```
GarbageCollected ComputedExpressionString::__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.

5.22.3.8 __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.

5.22.3.9 __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.

5.22.3.10 __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.

5.22.3.11 __not()

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

Compute the logical not of this value.

Returns

The result of the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.12 __string()

```
GarbageCollected ComputedExpressionString::__string ( ) const  [override], [virtual]
```

Perform a type cast to string.

Returns

The result of the the operation.

Reimplemented from Tang::ComputedExpression.

5.22.3.13 __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.

5.22.3.14 dump()

```
string ComputedExpressionString::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.

5.22.3.15 is_equal() [1/6]

```
bool ComputedExpressionString::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.

5.22.3.16 is_equal() [2/6]

```
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.

5.22.3.17 is_equal() [3/6]

```
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.

5.22.3.18 is_equal() [4/6]

```
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.

5.22.3.19 is_equal() [5/6]

```
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.

5.22.3.20 is_equal() [6/6]

```
bool ComputedExpressionString::is_equal (
            const string & 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.

5.22.3.21 makeCopy()

```
GarbageCollected ComputedExpressionString::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:

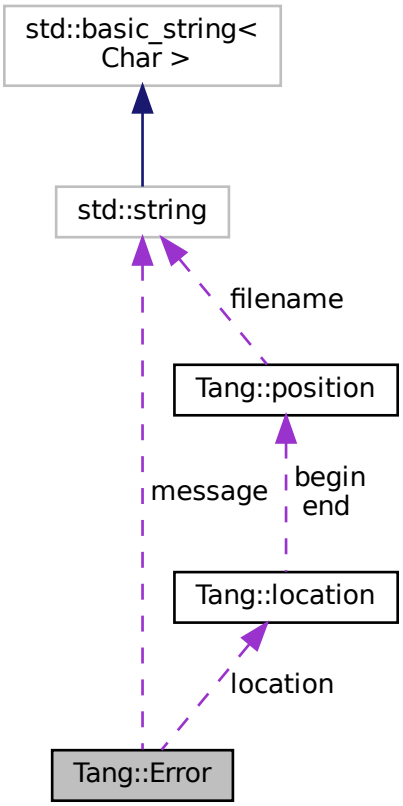
- include/computedExpressionString.hpp
- src/computedExpressionString.cpp

5.23 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.

5.23.1 Detailed Description

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

5.23.2 Constructor & Destructor Documentation

5.23.2.1 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.

5.23.2.2 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.

5.23.3 Friends And Related Function Documentation

5.23.3.1 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:

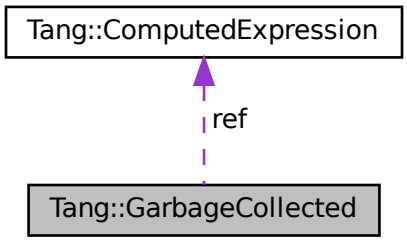
- include/error.hpp
- src/error.cpp

5.24 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 std::string &val) const

  Compare the GarbageCollected tracked object with a supplied value.
- bool operator== (const char *const &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.

5.24.1 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.

5.24.2 Constructor & Destructor Documentation

5.24.2.1 GarbageCollected() [1/3]

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

Copy Constructor.

Parameters

The	other GarbageCollected object to copy.

5.24.2.2 GarbageCollected() [2/3]

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

Move Constructor.

Parameters

The	other GarbageCollected object to move.

5.24.2.3 ~GarbageCollected()

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

Destructor.

Clean up the tracked object, if appropriate.

5.24.2.4 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.

5.24.3 Member Function Documentation

5.24.3.1 make()

```
template<class T , typename...  Args>
static GarbageCollected Tang::GarbageCollected::make (
```

```
            Args...  args )  [inline], [static]
```

Creates a garbage-collected object of the specified type.

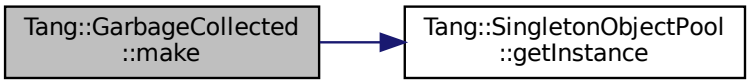
Parameters

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

Returns

A GarbageCollected object.

Here is the call graph for this function:



5.24.3.2 operator"!()

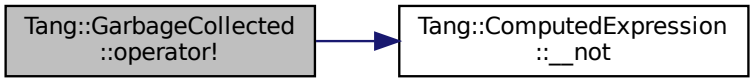
```
GarbageCollected GarbageCollected::operator!  ( ) const
```

Perform a logical not on the GarbageCollected value.

Returns

The result of the operation.

Here is the call graph for this function:



5.24.3.3 operator"!=()

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

Perform a != between two GarbageCollected values.

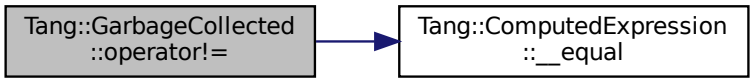
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.24.3.4 operator%()

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

Perform a modulo between two GarbageCollected values.

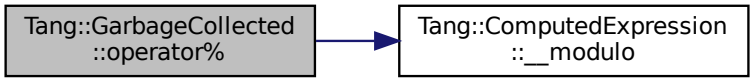
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.24.3.5 operator*() [1/2]

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

Access the tracked object.

Returns

A reference to the tracked object.

5.24.3.6 operator*() [2/2]

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

Perform a multiplication between two GarbageCollected values.

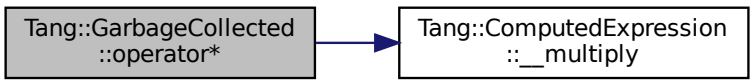
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.24.3.7 operator+()

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

Perform an addition between two GarbageCollected values.

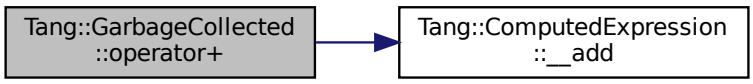
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.24.3.8 operator-() [1/2]

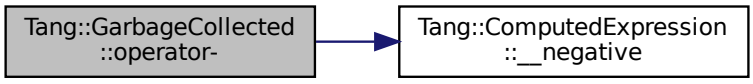
```
GarbageCollected GarbageCollected::operator- ( ) const
```

Perform a negation on the GarbageCollected value.

Returns

The result of the operation.

Here is the call graph for this function:



5.24.3.9 operator-() [2/2]

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

Perform a subtraction between two GarbageCollected values.

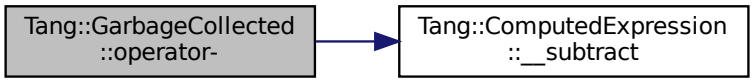
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.24.3.10 operator->()

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

Access the tracked object as a pointer.

Returns

A pointer to the tracked object.

5.24.3.11 operator/()

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

Perform a division between two GarbageCollected values.

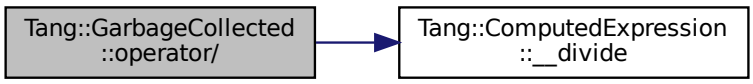
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.24.3.12 operator<()

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

Perform a < between two GarbageCollected values.

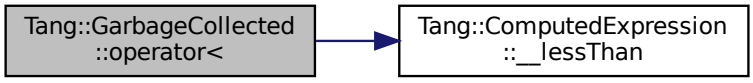
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.24.3.13 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.

5.24.3.14 operator=() [1/2]

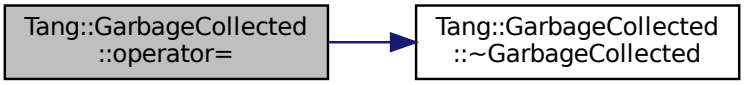
```
GarbageCollected& Tang::GarbageCollected::operator= (
            const GarbageCollected & other )  [inline]
```

Copy Assignment.

Parameters

The	other GarbageCollected object.

Here is the call graph for this function:



5.24.3.15 operator=() [2/2]

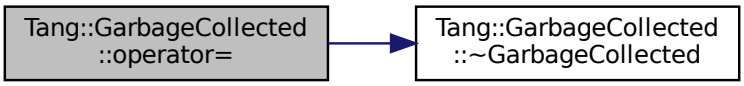
```
GarbageCollected& Tang::GarbageCollected::operator= (
            GarbageCollected && other )  [inline]
```

Move Assignment.

Parameters

The	other GarbageCollected object.

Here is the call graph for this function:



5.24.3.16 operator==() [1/8]

```
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.

5.24.3.17 operator==() [2/8]

```
bool GarbageCollected::operator== (
            const char *const & 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.

5.24.3.18 operator==() [3/8]

```
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.

5.24.3.19 operator==() [4/8]

```
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.

5.24.3.20 operator==() [5/8]

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

Perform a == between two GarbageCollected values.

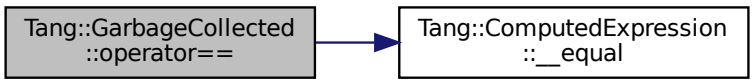
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.24.3.21 operator==() [6/8]

```
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.

5.24.3.22 operator==() [7/8]

```
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.

5.24.3.23 operator==() [8/8]

```
bool GarbageCollected::operator== (
            const std::string & 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.

5.24.3.24 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.

5.24.3.25 operator>=()

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

Perform a >= between two GarbageCollected values.

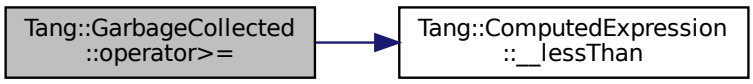
Parameters

rhs	The right hand side operand.

Returns

The result of the operation.

Here is the call graph for this function:



5.24.4 Friends And Related Function Documentation

5.24.4.1 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:

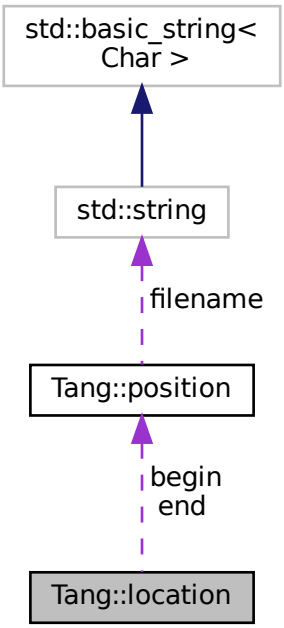
- include/garbageCollected.hpp
- src/garbageCollected.cpp

5.25 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.

5.25.1 Detailed Description

Two points in a source file.

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

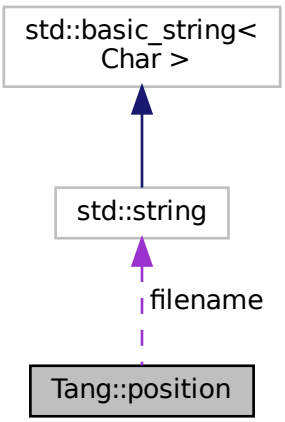
- build/generated/location.hh

5.26 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.

5.26.1 Detailed Description

A point in a source file.

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

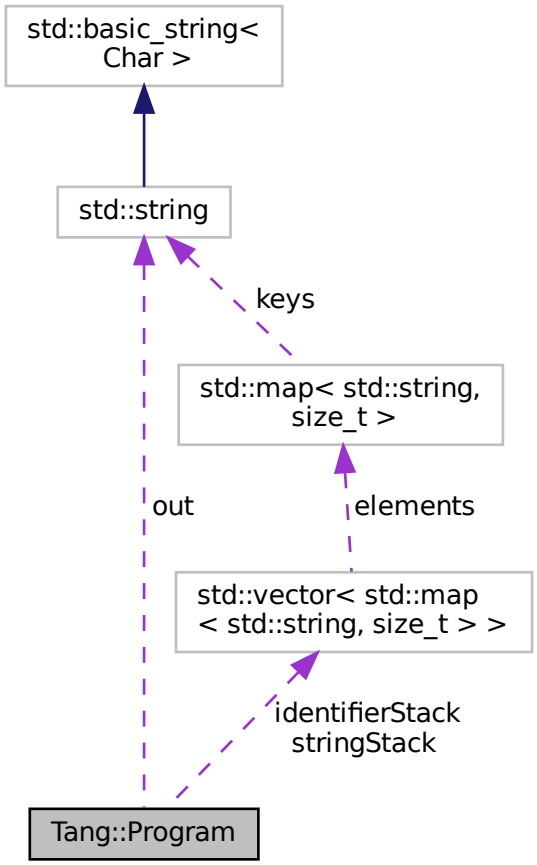
- build/generated/location.hh

5.27 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.
- std::vector< std::map< std::string, size_t > > stringStack

  Stack of mappings of strings to their stack locations.

5.27.1 Detailed Description

Represents a compiled script or template that may be executed.

5.27.2 Member Enumeration Documentation

5.27.2.1 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.

5.27.3 Constructor & Destructor Documentation

5.27.3.1 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`.

5.27.4 Member Function Documentation

5.27.4.1 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.

5.27.4.2 dumpBytecode()

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

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

Returns

A string containing the Opcode representation.

5.27.4.3 execute()

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

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

Returns

The current Program object.

5.27.4.4 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.

5.27.4.5 getBytecode()

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

Get the Bytecode vector.

Returns

The Bytecode vector.

5.27.4.6 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.

5.27.4.7 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.

5.27.4.8 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

5.28 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.

5.28.1 Detailed Description

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

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

5.28.2 Member Function Documentation

5.28.2.1 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.

5.28.2.2 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.

5.28.2.3 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

5.29 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.

5.29.1 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).

5.29.2 Constructor & Destructor Documentation

5.29.2.1 TangBase()

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

The constructor.

Isn't it glorious.

5.29.3 Member Function Documentation

5.29.3.1 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:

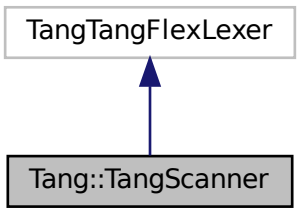
- include/tangBase.hpp
- src/tangBase.cpp

5.30 Tang::TangScanner Class Reference

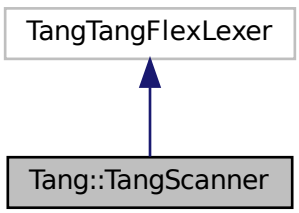
The Flex lexer class for the main Tang language.

`#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.

5.30.1 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.

5.30.2 Constructor & Destructor Documentation

5.30.2.1 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)

5.30.3 Member Function Documentation

5.30.3.1 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

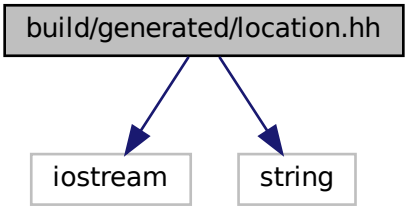
File Documentation

6.1 build/generated/location.hh File Reference

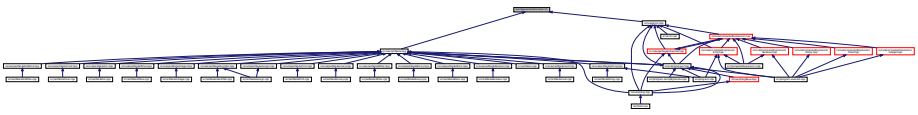
Define the Tang ::location class.

```
#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

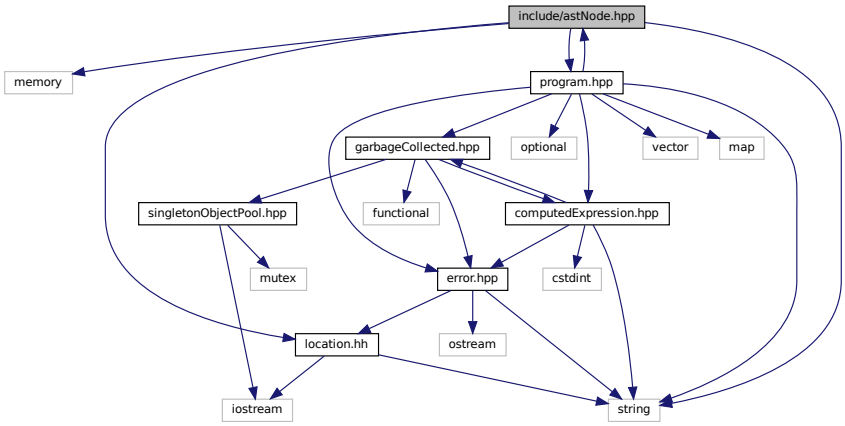
ostr	the destination output stream
pos	a reference to the position to redirect

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

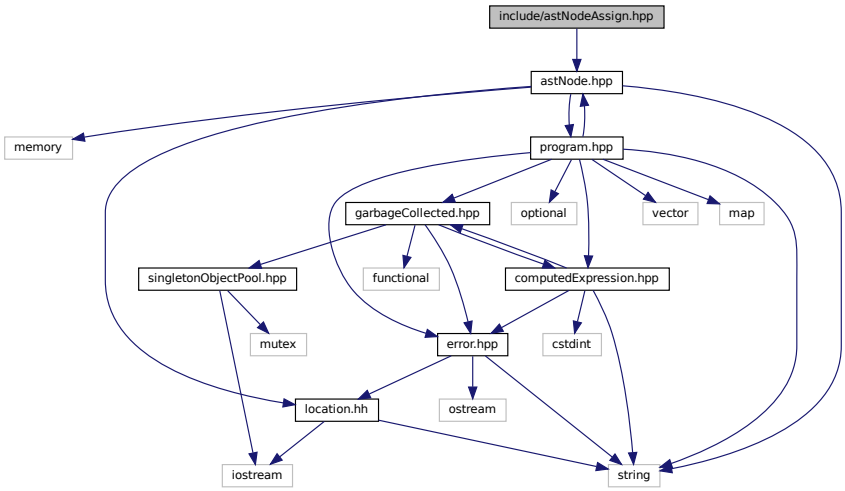
Declare the Tang::AstNode base class.

6.3 include/astNodeAssign.hpp File Reference

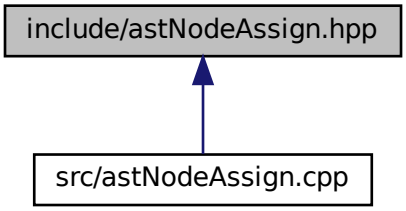
Declare the Tang::AstNodeAssign class.

```
#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

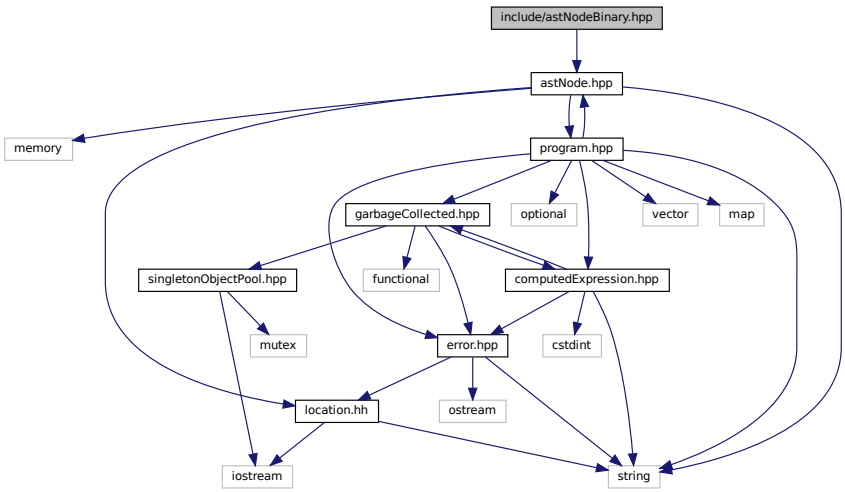
Declare the Tang::AstNodeAssign class.

6.4 include/astNodeBinary.hpp File Reference

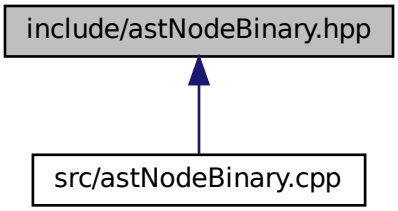
Declare the Tang::AstNodeBinary class.

```
#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

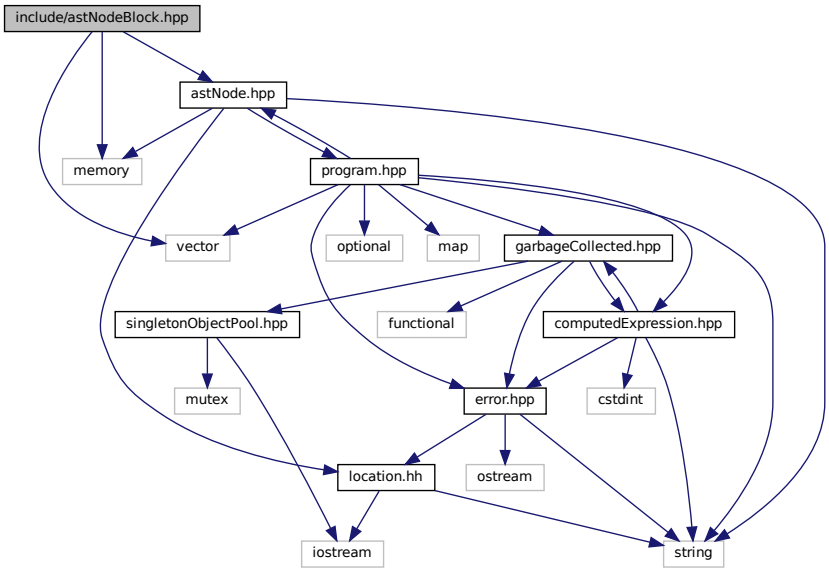
Declare the Tang::AstNodeBinary class.

6.5 include/astNodeBlock.hpp File Reference

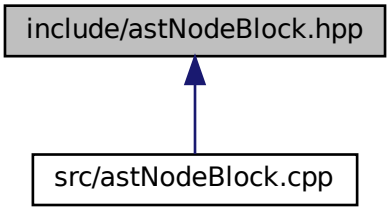
Declare the Tang::AstNodeBlock class.

```
#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

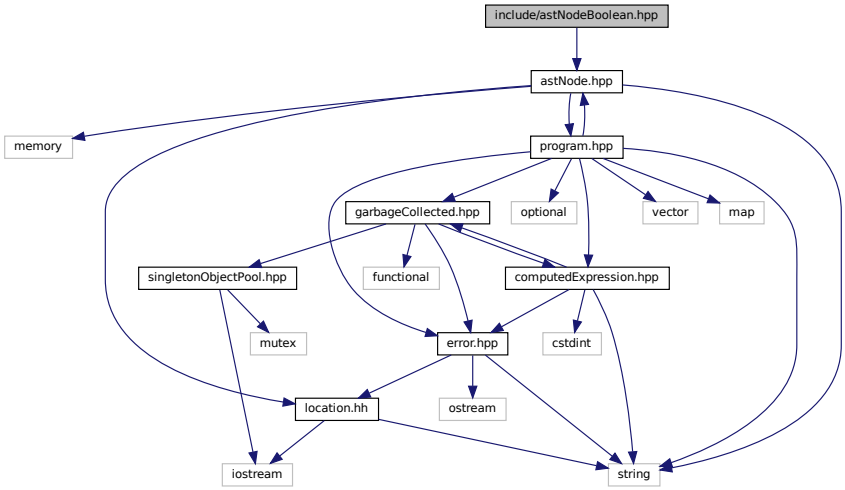
Declare the Tang::AstNodeBlock class.

6.6 include/astNodeBoolean.hpp File Reference

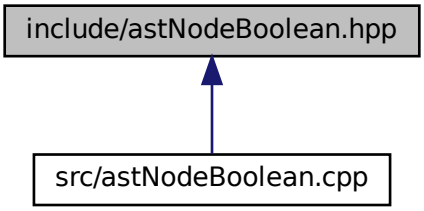
Declare the Tang::AstNodeBoolean class.

```
#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

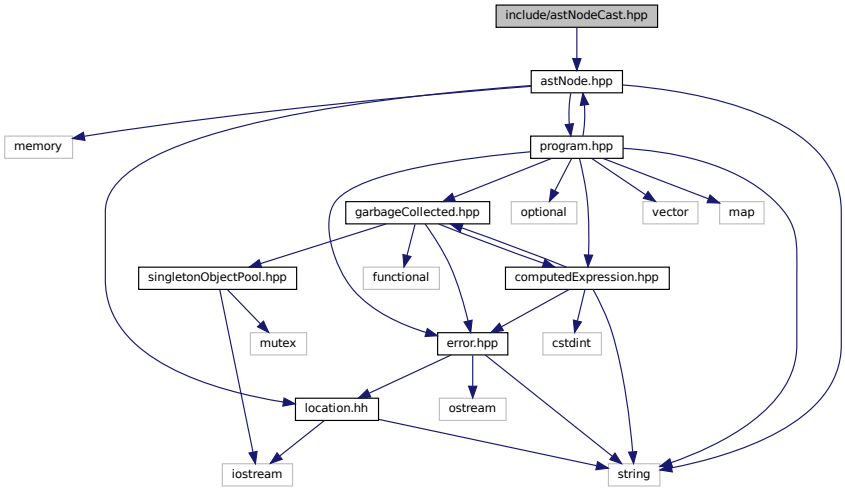
Declare the Tang::AstNodeBoolean class.

6.7 include/astNodeCast.hpp File Reference

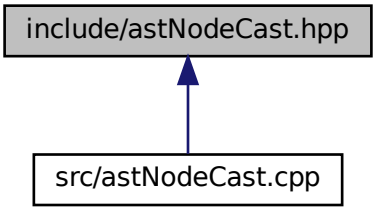
Declare the Tang::AstNodeCast class.

```
#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

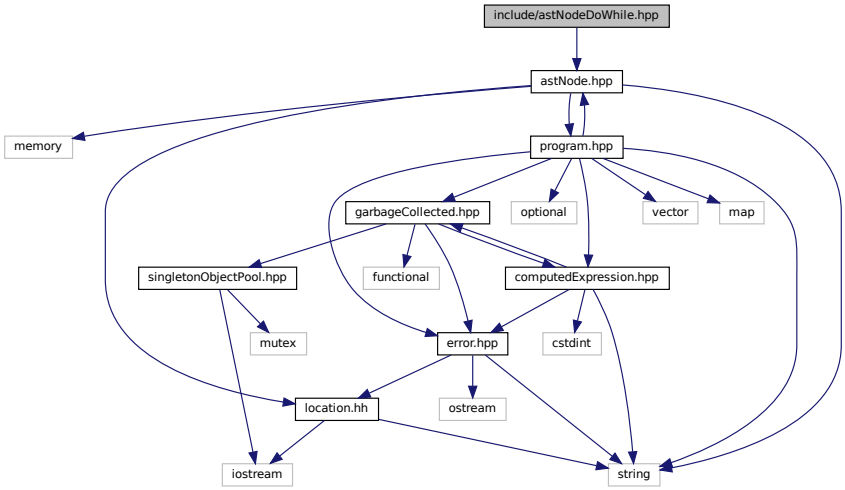
Declare the Tang::AstNodeCast class.

6.8 include/astNodeDoWhile.hpp File Reference

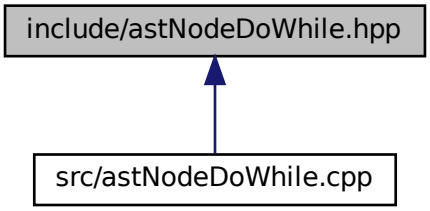
Declare the Tang::AstNodeDoWhile class.

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

Include dependency graph for astNodeDoWhile.hpp:



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



Classes

- class Tang::AstNodeDoWhile

  An AstNode that represents a do..while statement.

6.8.1 Detailed Description

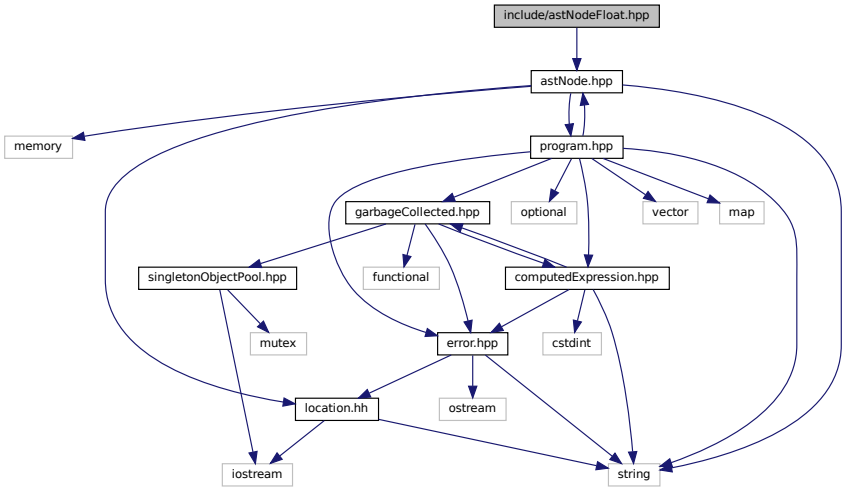
Declare the Tang::AstNodeDoWhile class.

6.9 include/astNodeFloat.hpp File Reference

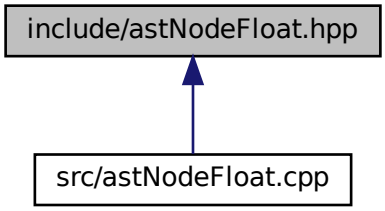
Declare the Tang::AstNodeFloat class.

```
#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.

6.9.1 Detailed Description

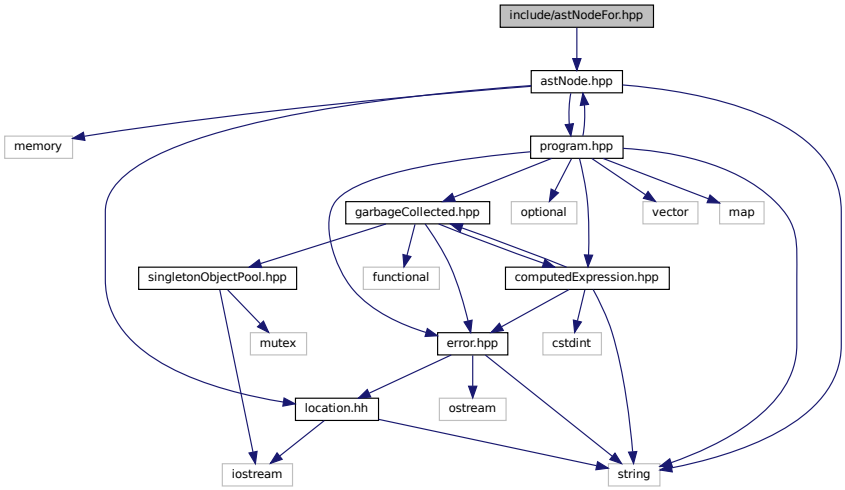
Declare the Tang::AstNodeFloat class.

6.10 include/astNodeFor.hpp File Reference

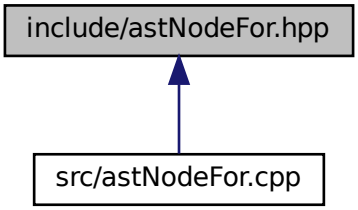
Declare the Tang::AstNodeFor class.

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

Include dependency graph for astNodeFor.hpp:



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



Classes

- class Tang::AstNodeFor

  An AstNode that represents an if() statement.

6.10.1 Detailed Description

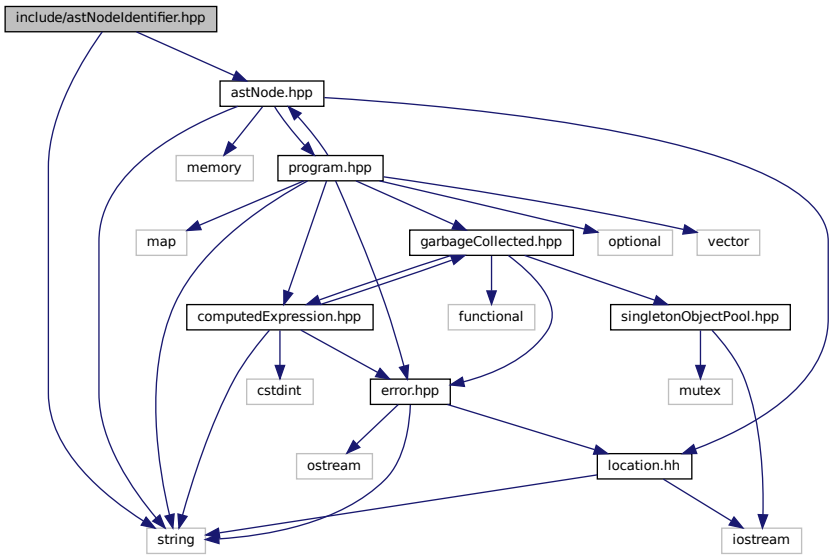
Declare the Tang::AstNodeFor class.

6.11 include/astNodeIdentifier.hpp File Reference

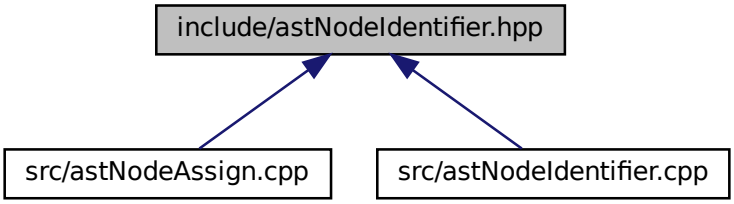
Declare the Tang::AstNodeIdentifier class.

```
#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.

6.11.1 Detailed Description

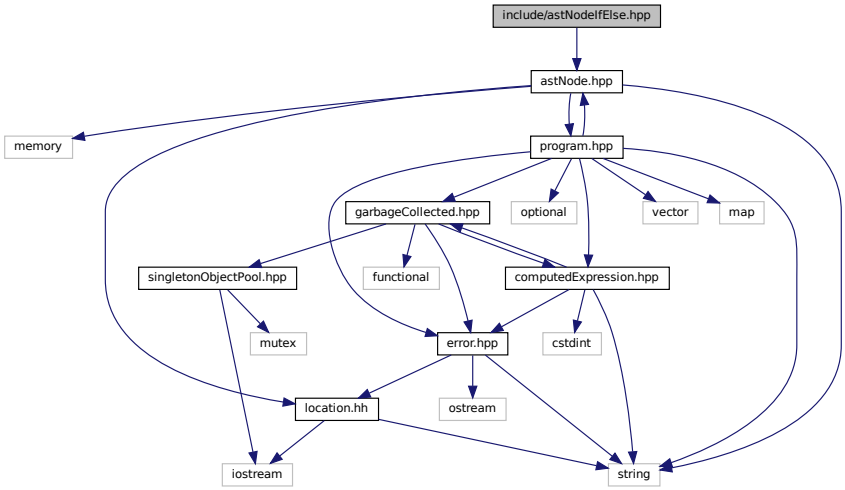
Declare the Tang::AstNodeIdentifier class.

6.12 include/astNodeIfElse.hpp File Reference

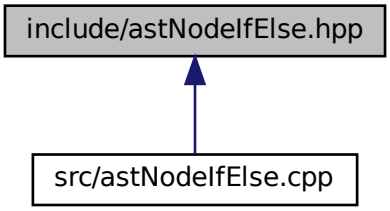
Declare the Tang::AstNodeIfElse class.

```
#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.

6.12.1 Detailed Description

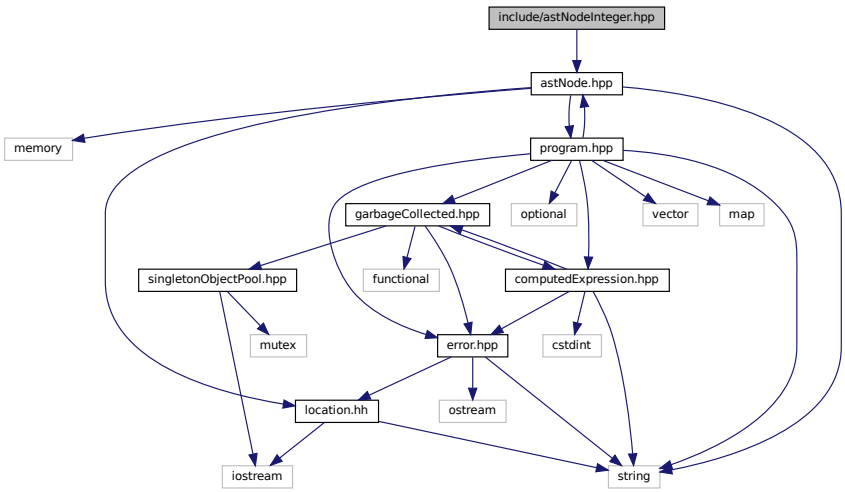
Declare the Tang::AstNodeIfElse class.

6.13 include/astNodeInteger.hpp File Reference

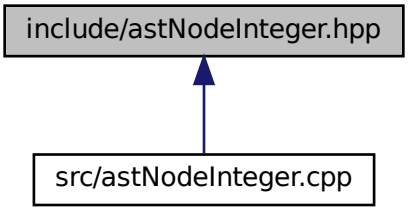
Declare the Tang::AstNodeInteger class.

```
#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.

6.13.1 Detailed Description

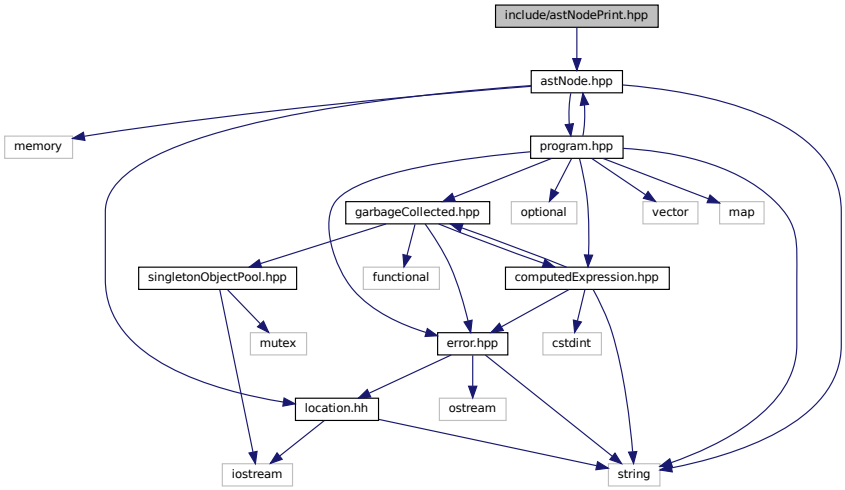
Declare the Tang::AstNodeInteger class.

6.14 include/astNodePrint.hpp File Reference

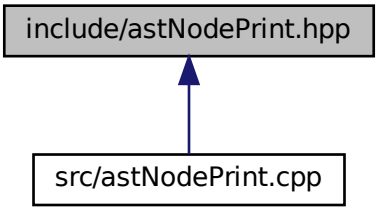
Declare the Tang::AstNodePrint class.

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

Include dependency graph for astNodePrint.hpp:



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



Classes

- class Tang::AstNodePrint

  An AstNode that represents a print typeeration.

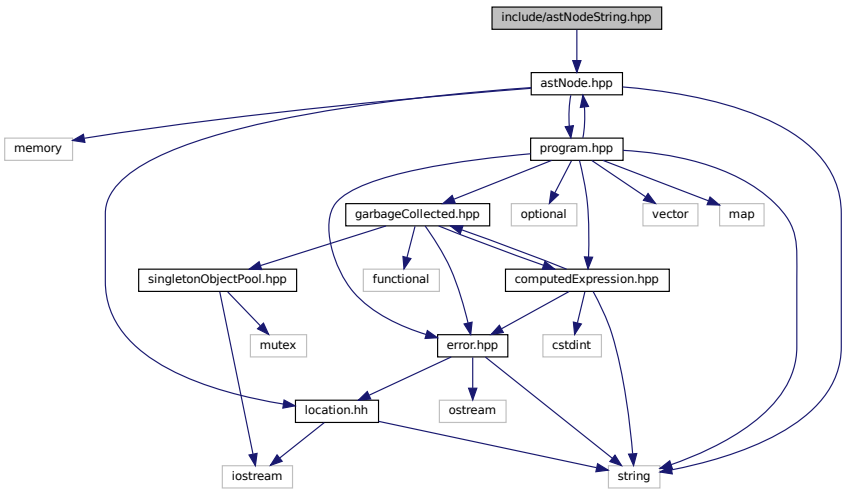
6.14.1 Detailed Description

Declare the Tang::AstNodePrint class.

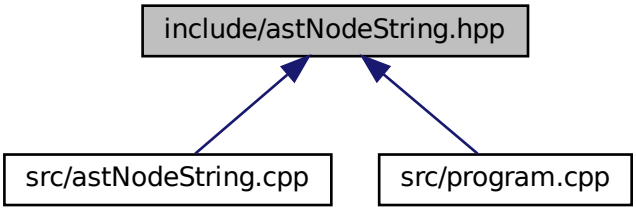
6.15 include/astNodeString.hpp File Reference

Declare the Tang::AstNodeString class.

`#include "astNode.hpp"`
Include dependency graph for astNodeString.hpp:



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



Classes

- class Tang::AstNodeString

  An AstNode that represents a string literal.

6.15.1 Detailed Description

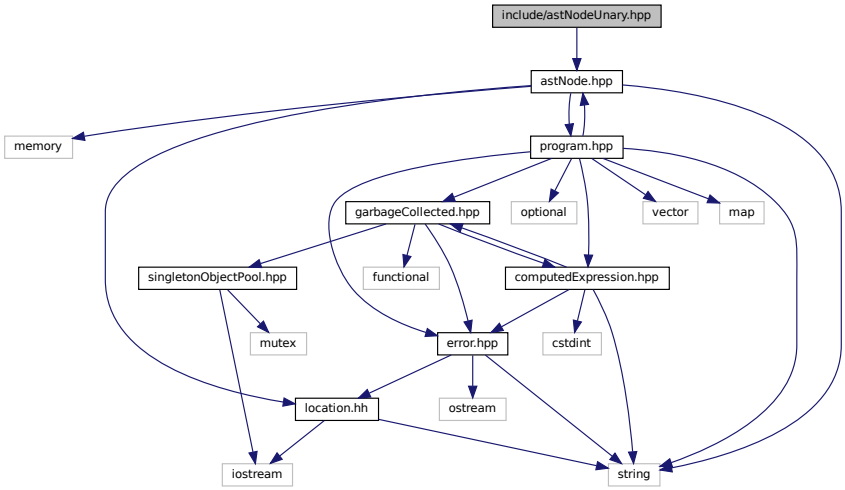
Declare the Tang::AstNodeString class.

6.16 include/astNodeUnary.hpp File Reference

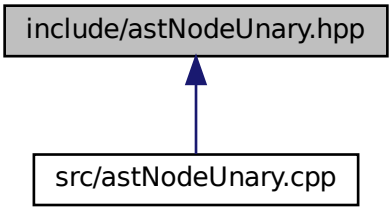
Declare the Tang::AstNodeUnary class.

```
#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.

6.16.1 Detailed Description

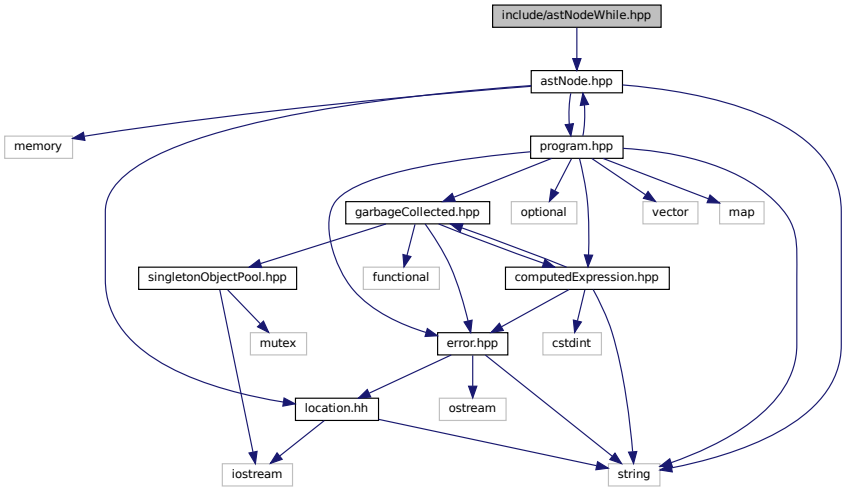
Declare the Tang::AstNodeUnary class.

6.17 include/astNodeWhile.hpp File Reference

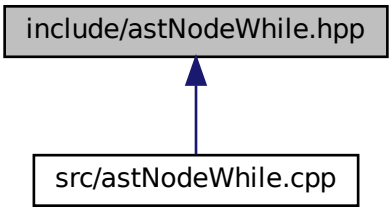
Declare the Tang::AstNodeWhile class.

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

Include dependency graph for astNodeWhile.hpp:



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



Classes

- class Tang::AstNodeWhile

  An AstNode that represents a while statement.

6.17.1 Detailed Description

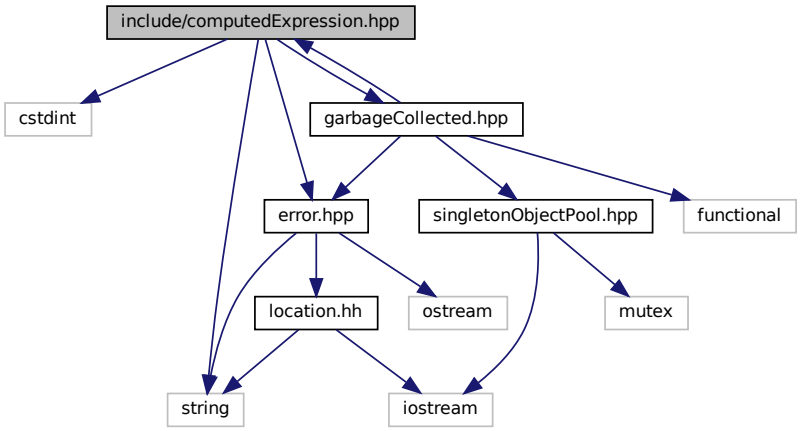
Declare the Tang::AstNodeWhile class.

6.18 include/computedExpression.hpp File Reference

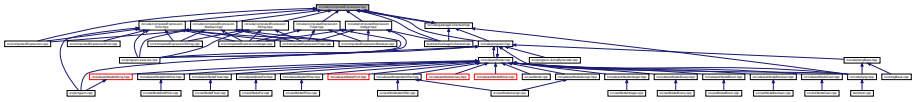
Declare the Tang::ComputedExpression base class.

```
#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.

6.18.1 Detailed Description

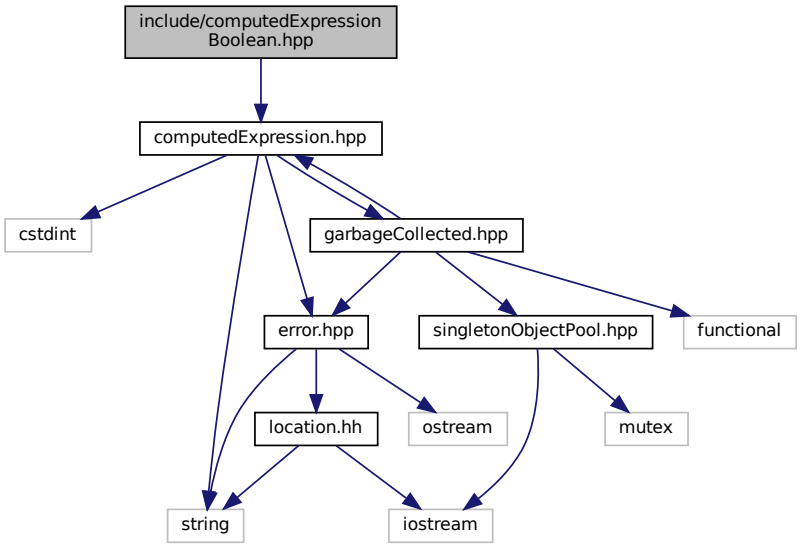
Declare the Tang::ComputedExpression base class.

6.19 include/computedExpressionBoolean.hpp File Reference

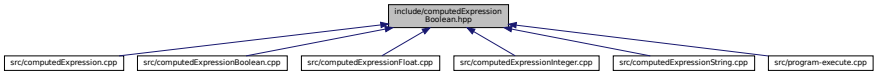
Declare the Tang::ComputedExpressionBoolean class.

```
#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.

6.19.1 Detailed Description

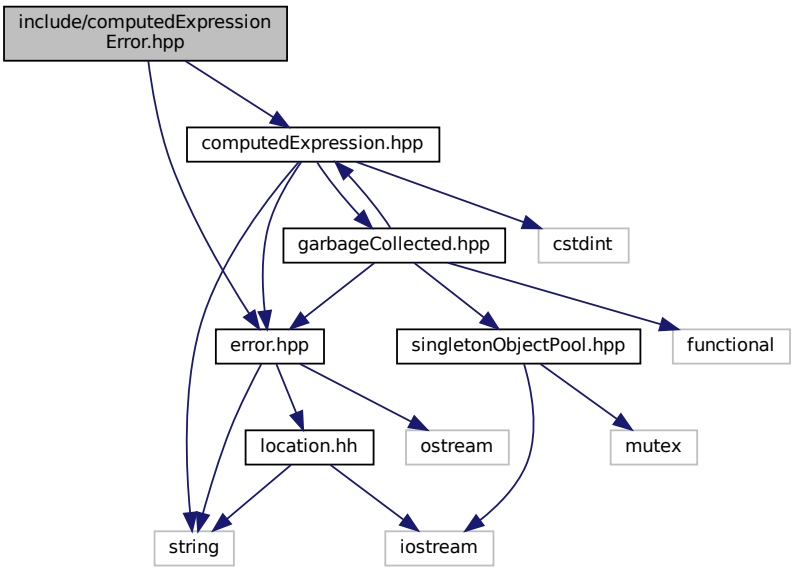
Declare the Tang::ComputedExpressionBoolean class.

6.20 include/computedExpressionError.hpp File Reference

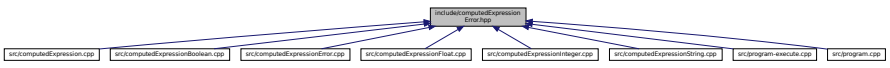
Declare the Tang::ComputedExpressionError class.

```
#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.

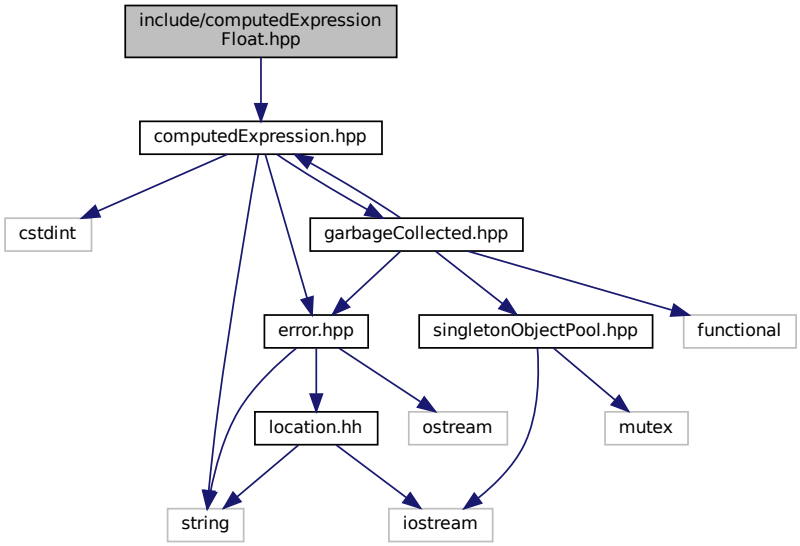
6.20.1 Detailed Description

Declare the Tang::ComputedExpressionError class.

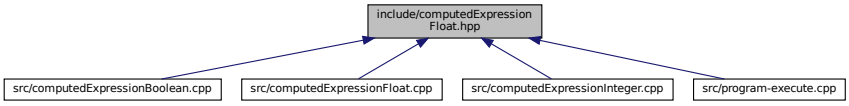
6.21 include/computedExpressionFloat.hpp File Reference

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.

6.21.1 Detailed Description

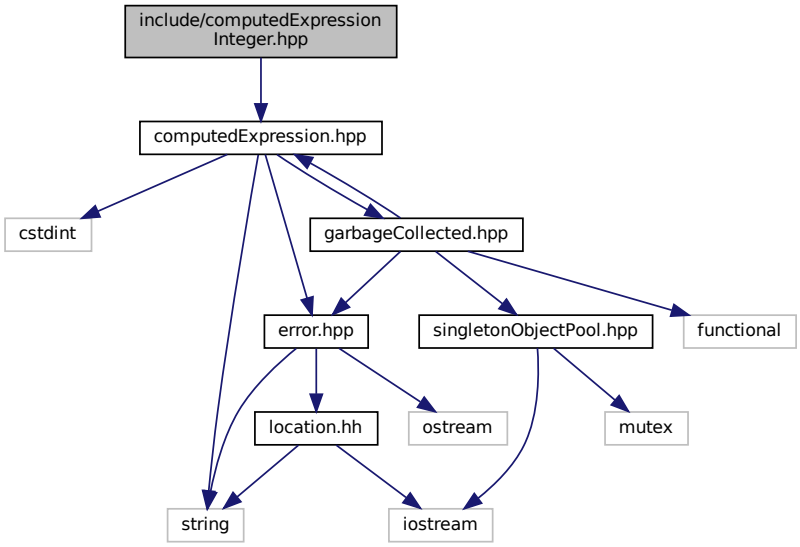
Declare the Tang::ComputedExpressionFloat class.

6.22 include/computedExpressionInteger.hpp File Reference

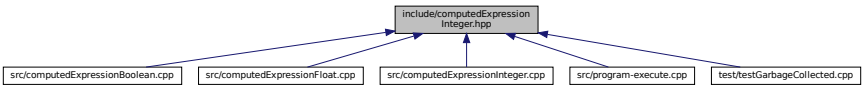
Declare the Tang::ComputedExpressionInteger class.

```
#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.

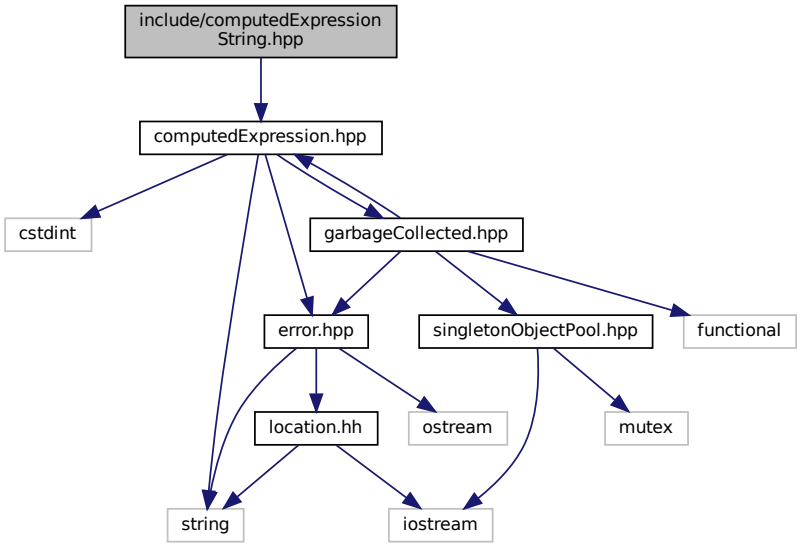
6.22.1 Detailed Description

Declare the Tang::ComputedExpressionInteger class.

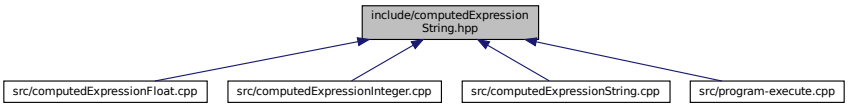
6.23 include/computedExpressionString.hpp File Reference

Declare the Tang::ComputedExpressionString class.

`#include "computedExpression.hpp"`
Include dependency graph for computedExpressionString.hpp:



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



Classes

- class Tang::ComputedExpressionString

  Represents a String that is the result of a computation.

6.23.1 Detailed Description

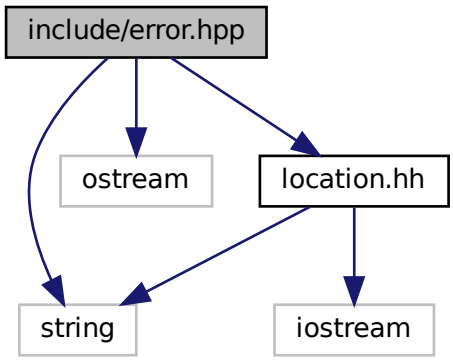
Declare the Tang::ComputedExpressionString class.

6.24 include/error.hpp File Reference

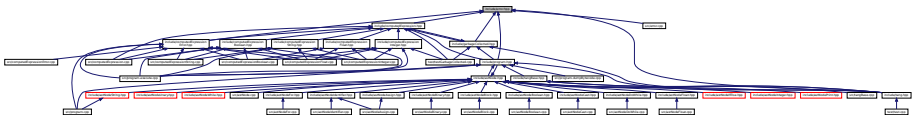
Declare the Tang::Error class used to describe syntax and runtime errors.

```
#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.

6.24.1 Detailed Description

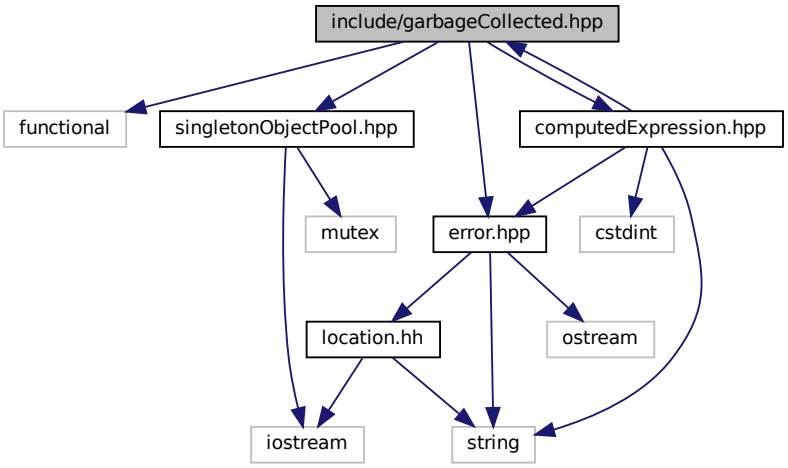
Declare the Tang::Error class used to describe syntax and runtime errors.

6.25 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.

6.25.1 Detailed Description

Declare the Tang::GarbageCollected class.

6.26 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.

6.26.1 Detailed Description

Contains generic macros.

6.26.2 Macro Definition Documentation

6.26.2.1 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.

6.27 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 , JMPF_POP , JMPT , JMPT_POP ,
  NULLVAL , INTEGER , FLOAT , BOOLEAN ,
  STRING , ADD , SUBTRACT , MULTIPLY ,
  DIVIDE , MODULO , NEGATIVE , NOT ,
  LT , LTE , GT , GTE ,
  EQ , NEQ , CASTINTEGER , CASTFLOAT ,
  CASTBOOLEAN , PRINT }

6.27.1 Detailed Description

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

6.27.2 Enumeration Type Documentation

6.27.2.1 Opcode

`enum Tang::Opcode [strong]`

Enumerator

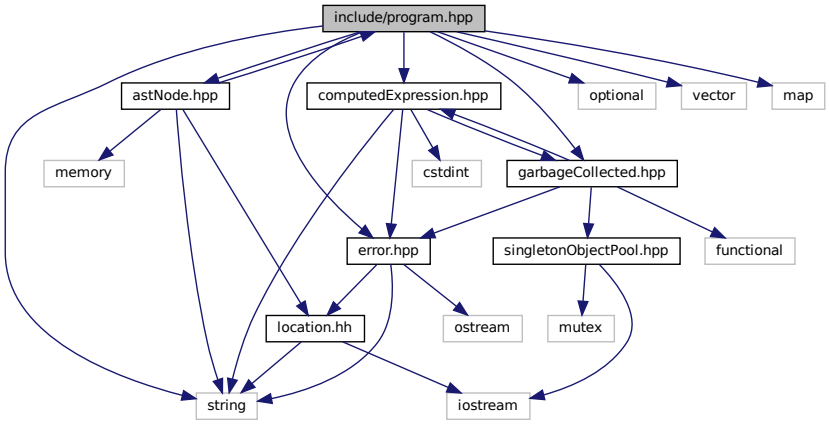
POP	Pop a val.
PEEK	Stack # (from fp): push val from stack #.
POKE	Stack # (from fp): Copy a val, store @ stack #.
JMP	PC #: set pc to PC #.
JMPF	PC #: read val, if false, set pc to PC #.
JMPF_POP	PC #: pop val, if false, set pc to PC #.
JMPT	PC #: read val, if true, set pc to PC #.
JMPT_POP	PC #: pop val, if true, 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.
STRING	Get len, char string: push string.
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.
PRINT	Pop val, print(val), push error or NULL.

6.28 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.

6.28.1 Detailed Description

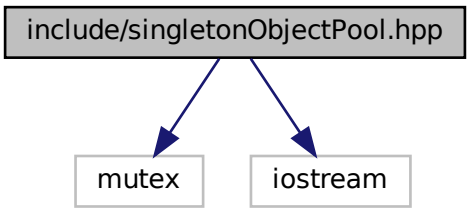
Declare the Tang::Program class used to compile and execute source code.

6.29 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.

6.29.1 Detailed Description

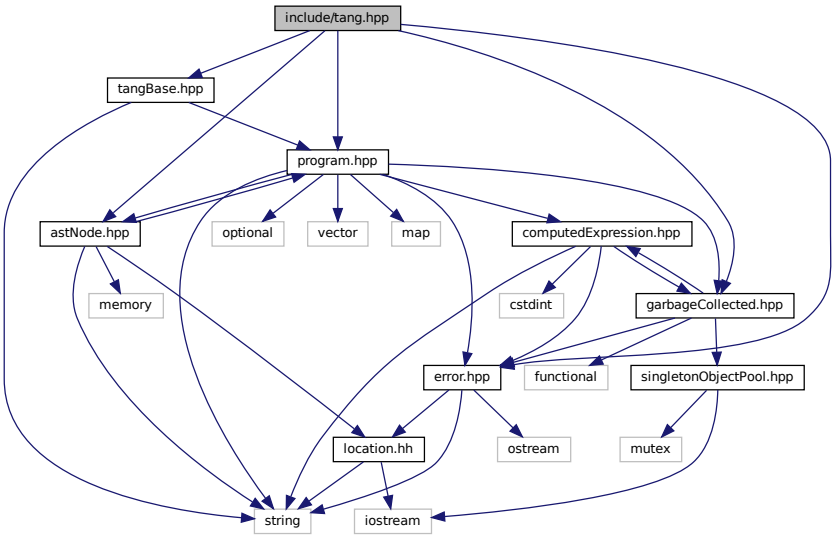
Declare the Tang::SingletonObjectPool class.

6.30 include/tang.hpp File Reference

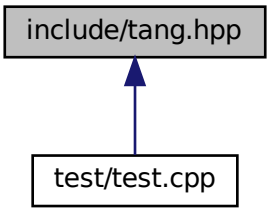
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

```
#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:



6.30.1 Detailed Description

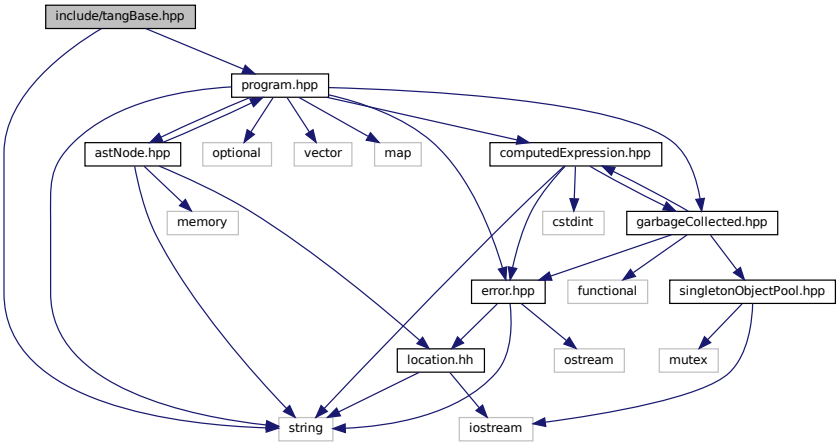
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

6.31 include/tangBase.hpp File Reference

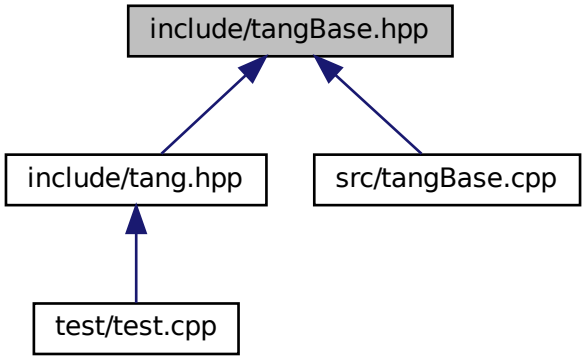
Declare the Tang::TangBase class used to interact with Tang.

```
#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.

6.31.1 Detailed Description

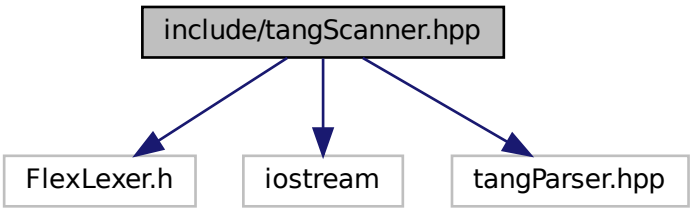
Declare the Tang::TangBase class used to interact with Tang.

6.32 include/tangScanner.hpp File Reference

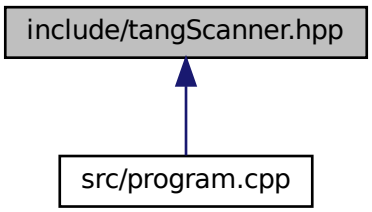
Declare the Tang::TangScanner used to tokenize a Tang script.

```
#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()

6.32.1 Detailed Description

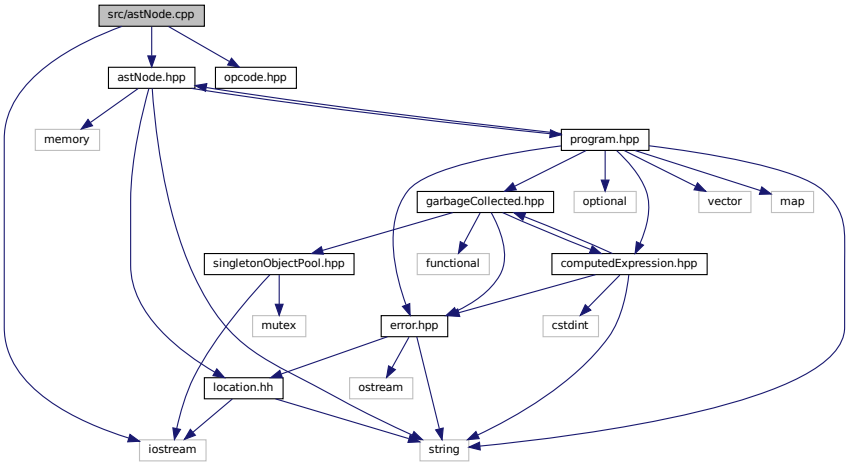
Declare the Tang::TangScanner used to tokenize a Tang script.

6.33 src/astNode.cpp File Reference

Define the Tang::AstNode class.

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

Include dependency graph for astNode.cpp:



6.33.1 Detailed Description

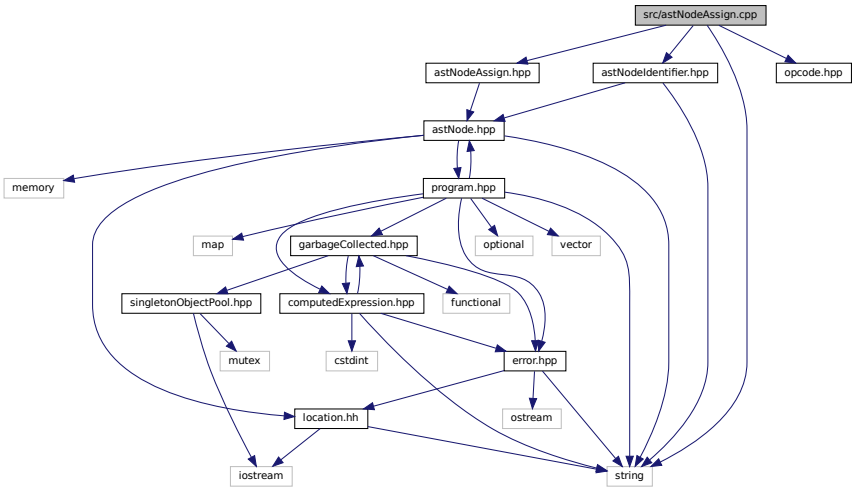
Define the Tang::AstNode class.

6.34 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:



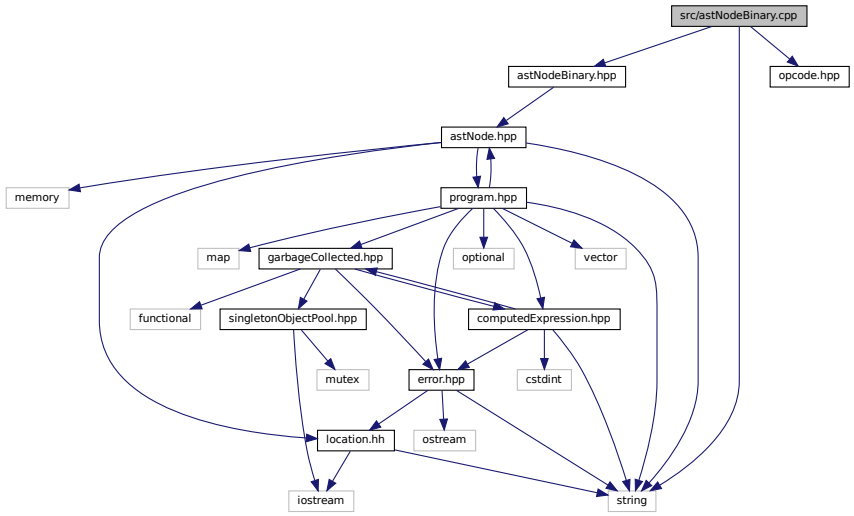
6.34.1 Detailed Description

Define the Tang::AstNodeAssign class.

6.35 src/astNodeBinary.cpp File Reference

Define the Tang::AstNodeBinary class.

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



6.35.1 Detailed Description

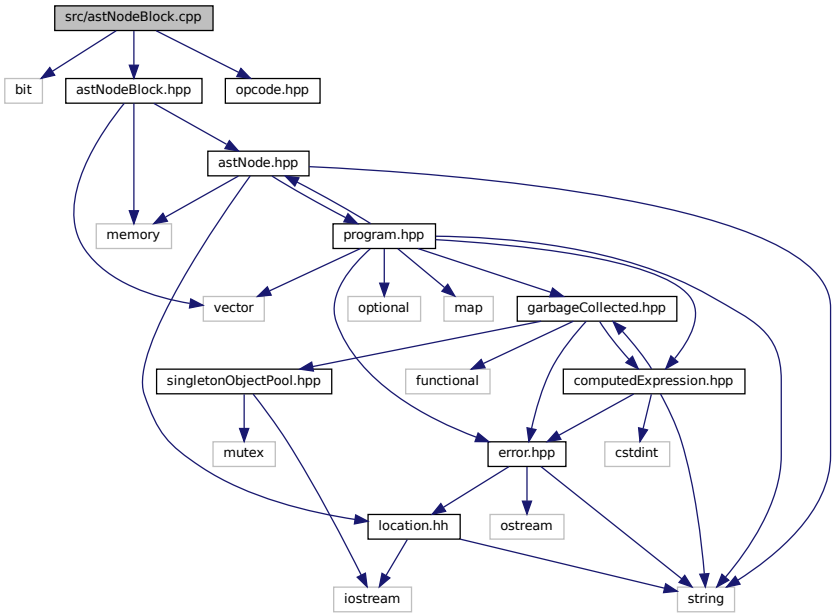
Define the Tang::AstNodeBinary class.

6.36 src/astNodeBlock.cpp File Reference

Define the Tang::AstNodeBlock class.

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

Include dependency graph for astNodeBlock.cpp:



6.36.1 Detailed Description

Define the Tang::AstNodeBlock class.

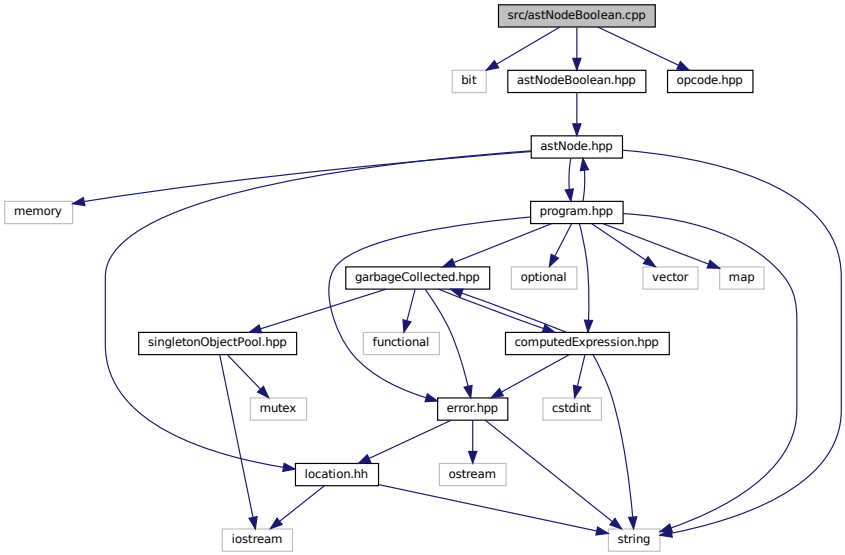
6.37 src/astNodeBoolean.cpp File Reference

Define the Tang::AstNodeBoolean class.

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

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

Include dependency graph for astNodeBoolean.cpp:



6.37.1 Detailed Description

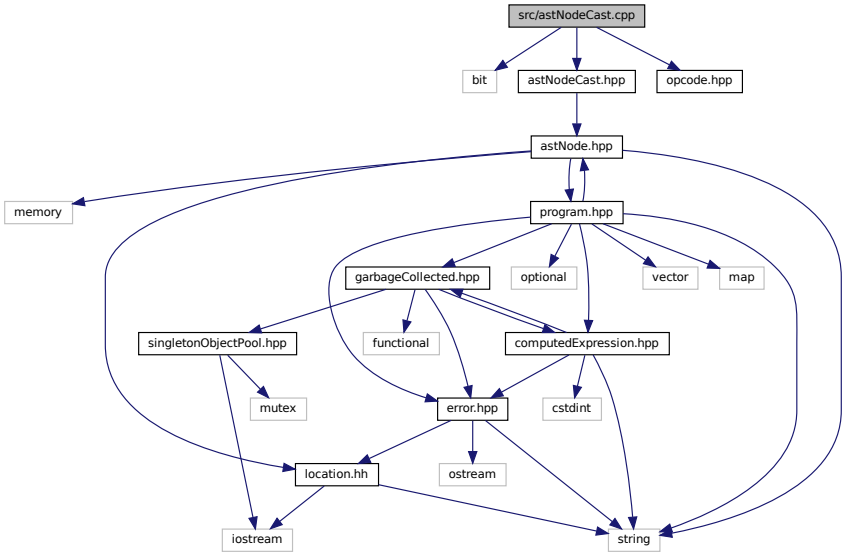
Define the Tang::AstNodeBoolean class.

6.38 src/astNodeCast.cpp File Reference

Define the Tang::AstNodeCast class.

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

Include dependency graph for astNodeCast.cpp:



6.38.1 Detailed Description

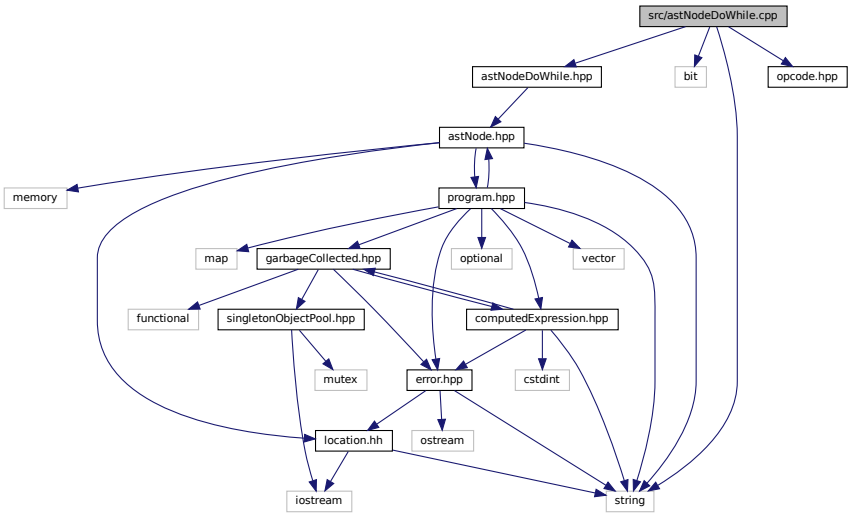
Define the Tang::AstNodeCast class.

6.39 src/astNodeDoWhile.cpp File Reference

Define the Tang::AstNodeDoWhile class.

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

Include dependency graph for astNodeDoWhile.cpp:



6.39.1 Detailed Description

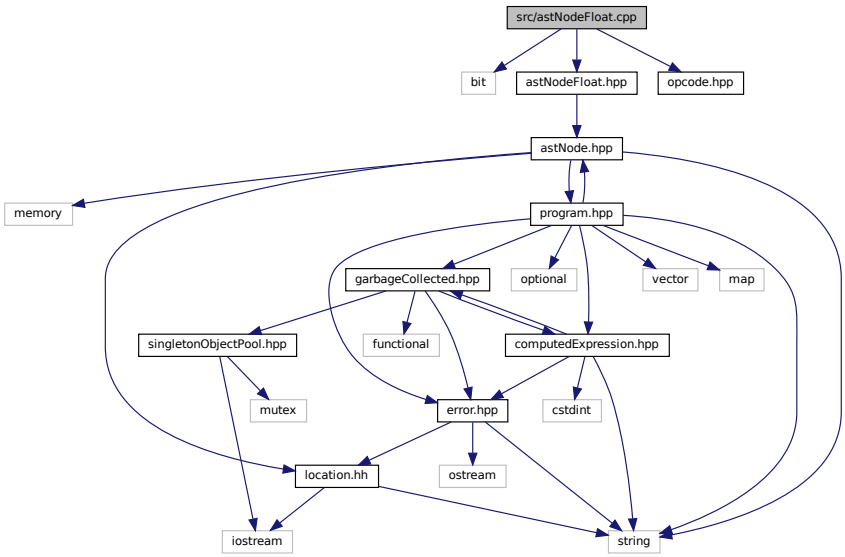
Define the Tang::AstNodeDoWhile class.

6.40 src/astNodeFloat.cpp File Reference

Define the Tang::AstNodeFloat class.

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

Include dependency graph for astNodeFloat.cpp:



6.40.1 Detailed Description

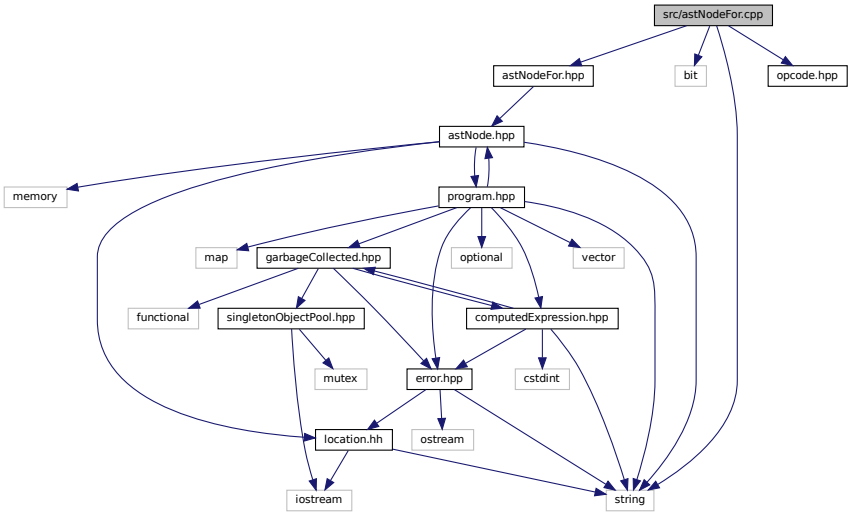
Define the Tang::AstNodeFloat class.

6.41 src/astNodeFor.cpp File Reference

Define the Tang::AstNodeFor class.

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

Include dependency graph for astNodeFor.cpp:



6.41.1 Detailed Description

Define the Tang::AstNodeFor class.

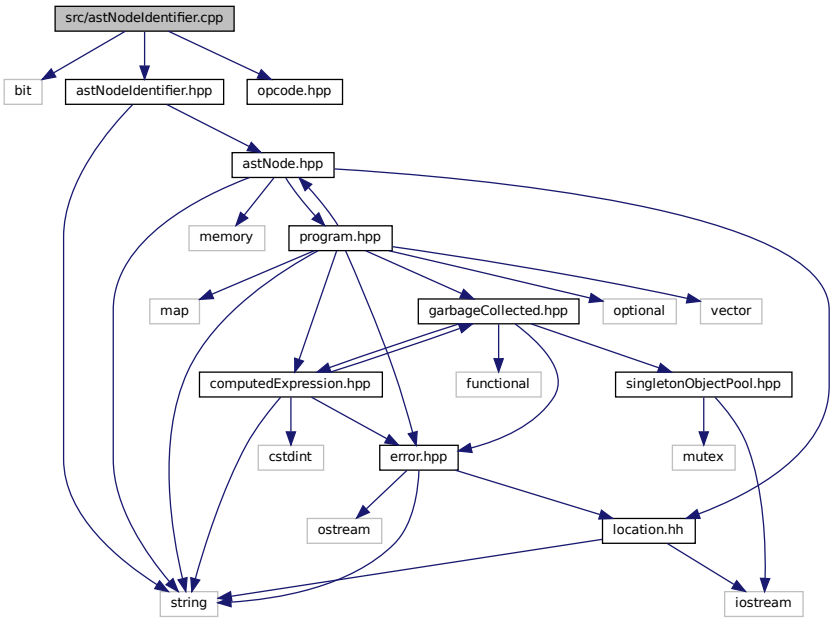
6.42 src/astNodeIdentifier.cpp File Reference

Define the Tang::AstNodeIdentifier class.

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

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

Include dependency graph for astNodeIdentifier.cpp:



6.42.1 Detailed Description

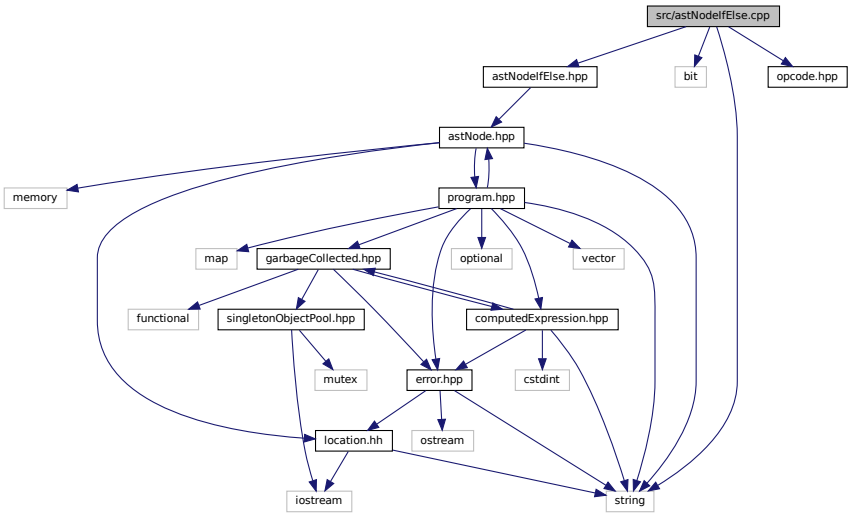
Define the Tang::AstNodeIdentifier class.

6.43 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:



6.43.1 Detailed Description

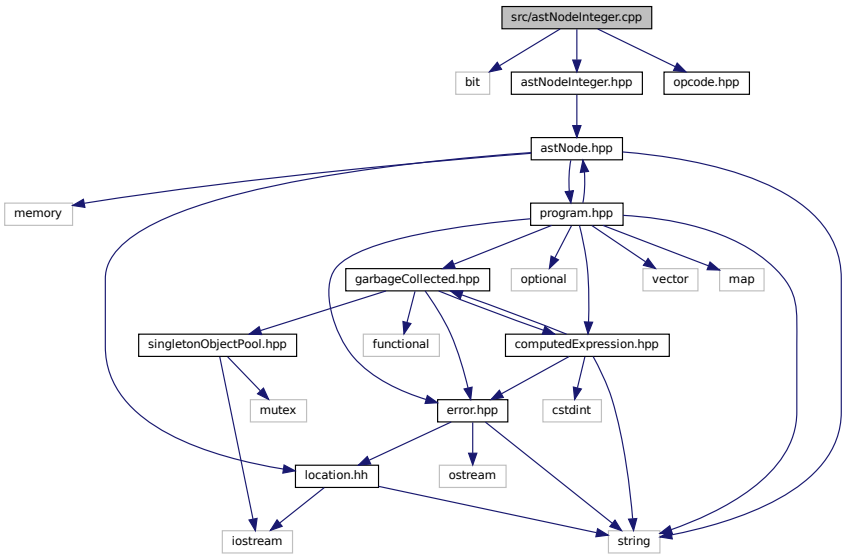
Define the Tang::AstNodeIfElse class.

6.44 src/astNodeInteger.cpp File Reference

Define the Tang::AstNodeInteger class.

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

Include dependency graph for astNodeInteger.cpp:



6.44.1 Detailed Description

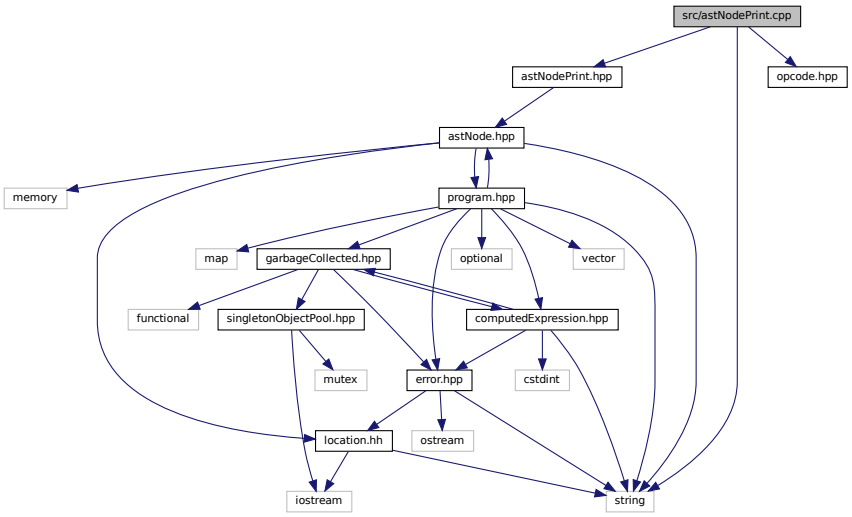
Define the Tang::AstNodeInteger class.

6.45 src/astNodePrint.cpp File Reference

Define the Tang::AstNodePrint class.

```
#include <string>
#include "astNodePrint.hpp"
#include "opcode.hpp"
```

Include dependency graph for astNodePrint.cpp:



6.45.1 Detailed Description

Define the Tang::AstNodePrint class.

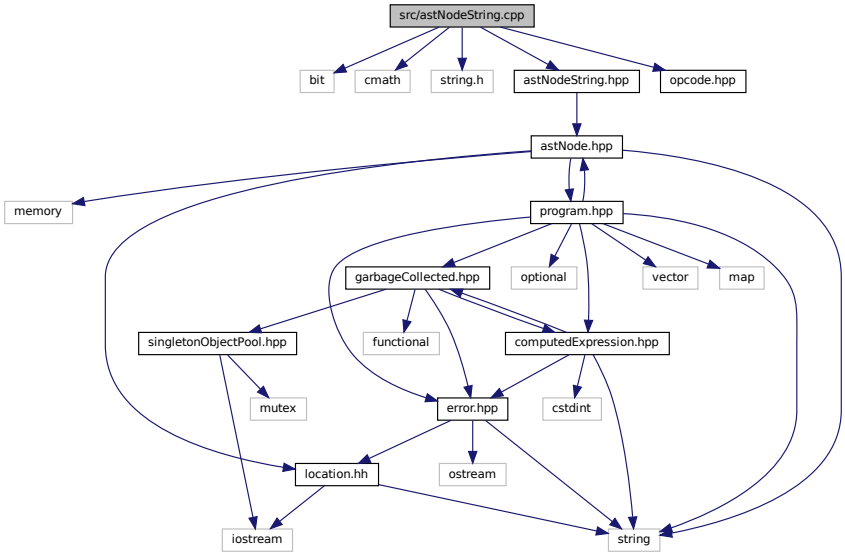
6.46 src/astNodeString.cpp File Reference

Define the Tang::AstNodeString class.

```
#include <bit>
#include <cmath>
#include <string.h>
#include "astNodeString.hpp"
```

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

Include dependency graph for astNodeString.cpp:



6.46.1 Detailed Description

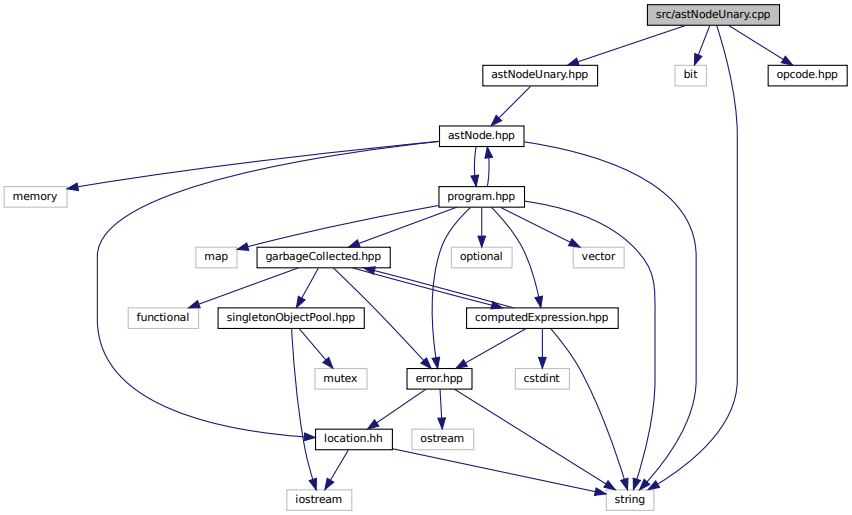
Define the Tang::AstNodeString class.

6.47 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:



6.47.1 Detailed Description

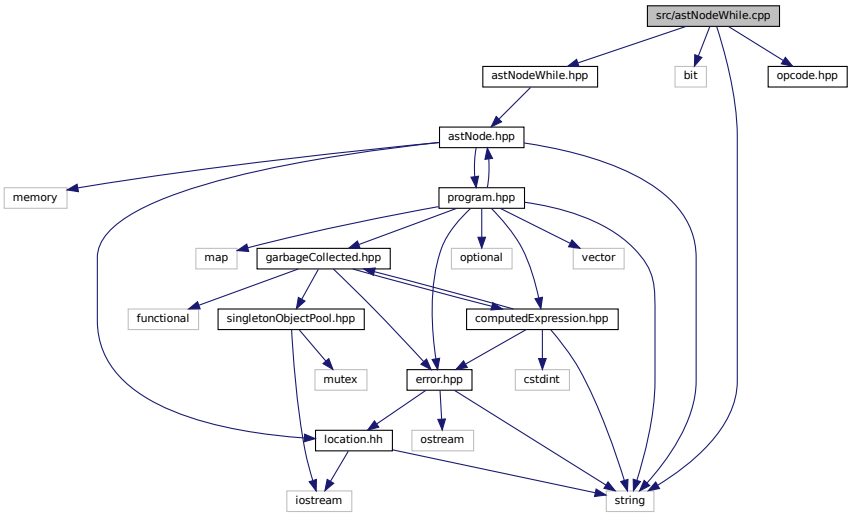
Define the Tang::AstNodeUnary class.

6.48 src/astNodeWhile.cpp File Reference

Define the Tang::AstNodeWhile class.

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

Include dependency graph for astNodeWhile.cpp:



6.48.1 Detailed Description

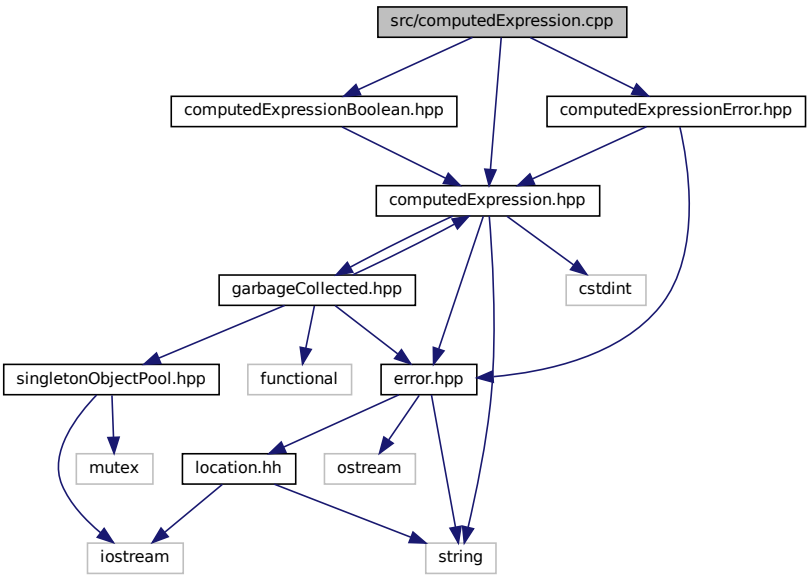
Define the Tang::AstNodeWhile class.

6.49 src/computedExpression.cpp File Reference

Define the Tang::ComputedExpression class.

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

Include dependency graph for computedExpression.cpp:



6.49.1 Detailed Description

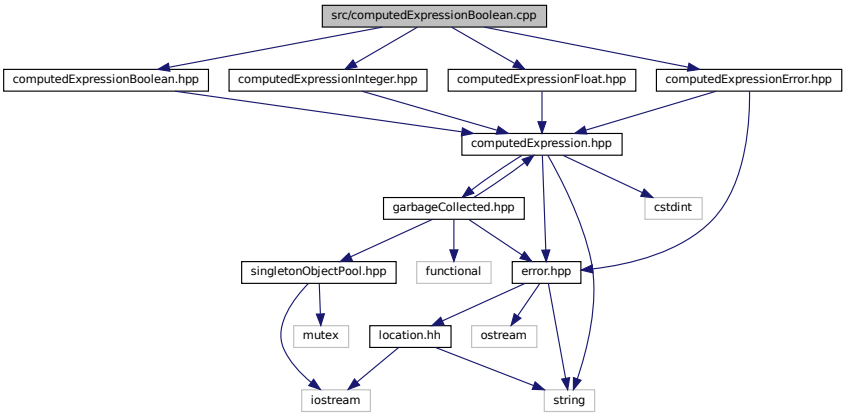
Define the Tang::ComputedExpression class.

6.50 src/computedExpressionBoolean.cpp File Reference

Define the Tang::ComputedExpressionBoolean class.

```
#include "computedExpressionBoolean.hpp"
#include "computedExpressionInteger.hpp"
#include "computedExpressionFloat.hpp"
```

```
#include "computedExpressionError.hpp"
```
Include dependency graph for computedExpressionBoolean.cpp:



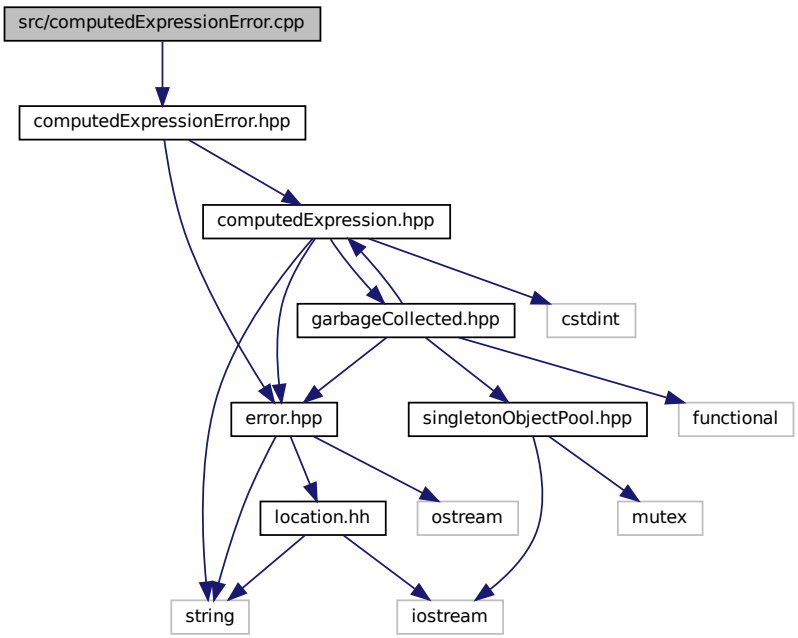
6.50.1 Detailed Description

Define the Tang::ComputedExpressionBoolean class.

6.51 src/computedExpressionError.cpp File Reference

Define the Tang::ComputedExpressionError class.

```
#include "computedExpressionError.hpp"
```
Include dependency graph for computedExpressionError.cpp:



6.51.1 Detailed Description

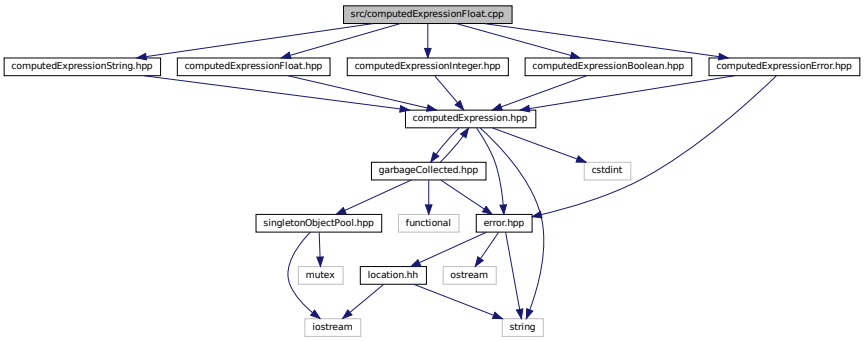
Define the Tang::ComputedExpressionError class.

6.52 src/computedExpressionFloat.cpp File Reference

Define the Tang::ComputedExpressionFloat class.

```
#include "computedExpressionFloat.hpp"
#include "computedExpressionInteger.hpp"
#include "computedExpressionBoolean.hpp"
#include "computedExpressionString.hpp"
#include "computedExpressionError.hpp"
```

Include dependency graph for computedExpressionFloat.cpp:



6.52.1 Detailed Description

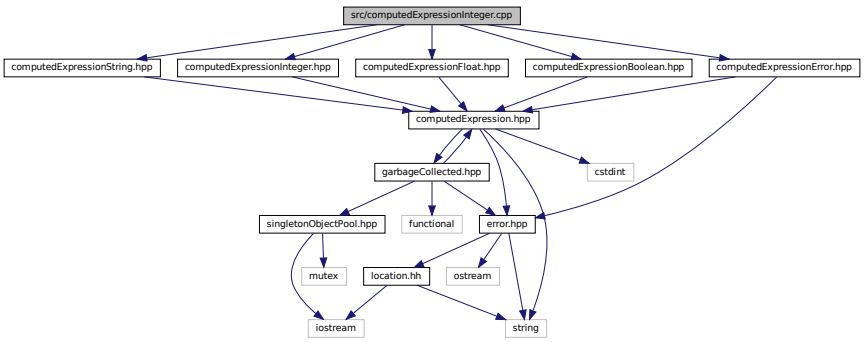
Define the Tang::ComputedExpressionFloat class.

6.53 src/computedExpressionInteger.cpp File Reference

Define the Tang::ComputedExpressionInteger class.

```
#include "computedExpressionInteger.hpp"
#include "computedExpressionFloat.hpp"
#include "computedExpressionBoolean.hpp"
#include "computedExpressionString.hpp"
```

```
#include "computedExpressionError.hpp"
```
Include dependency graph for computedExpressionInteger.cpp:



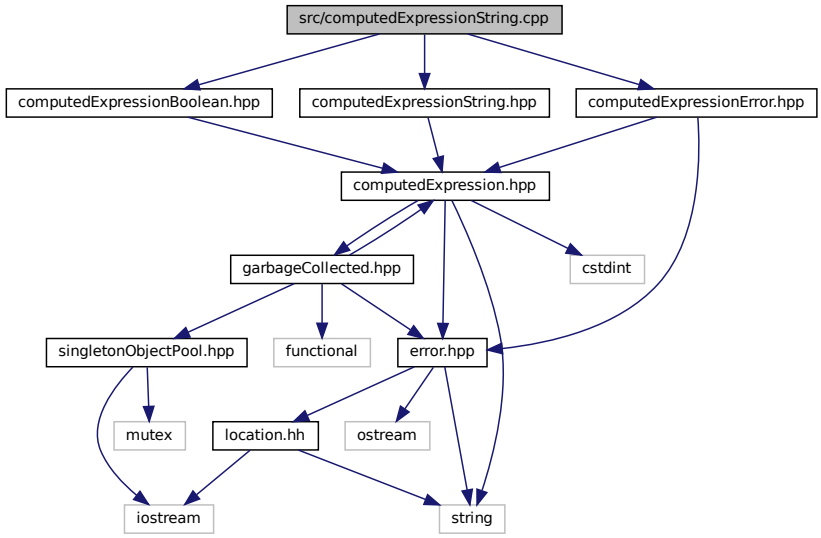
6.53.1 Detailed Description

Define the Tang::ComputedExpressionInteger class.

6.54 src/computedExpressionString.cpp File Reference

Define the Tang::ComputedExpressionString class.

```
#include "computedExpressionString.hpp"
#include "computedExpressionBoolean.hpp"
#include "computedExpressionError.hpp"
```
Include dependency graph for computedExpressionString.cpp:



6.54.1 Detailed Description

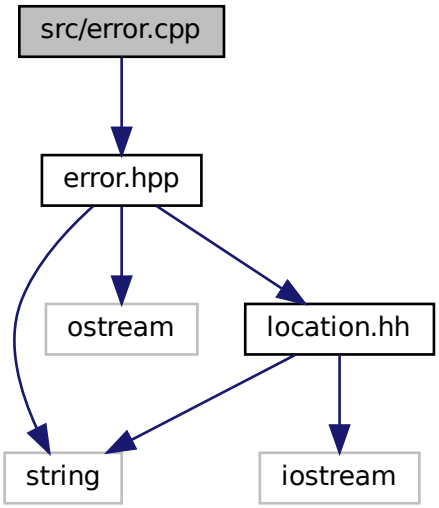
Define the Tang::ComputedExpressionString class.

6.55 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)

6.55.1 Detailed Description

Define the Tang::Error class.

6.55.2 Function Documentation

6.55.2.1 operator<<()

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

Parameters

out	The output stream.
error	The Error object.

Returns

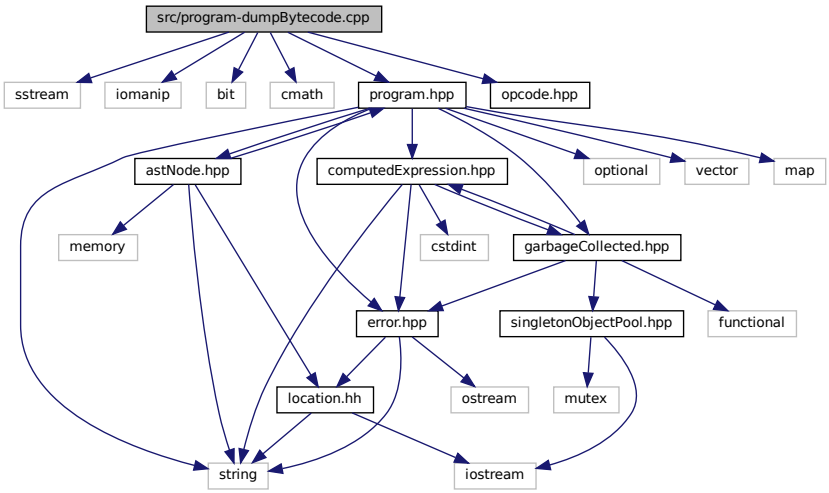
The output stream.

6.56 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.

6.56.1 Detailed Description

Define the Tang::Program::dumpBytecode method.

6.56.2 Macro Definition Documentation

6.56.2.1 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

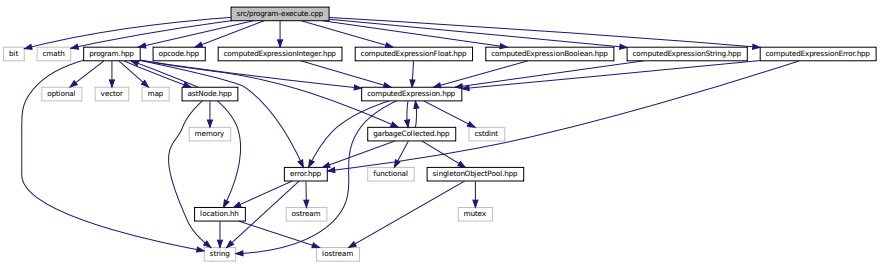
x	The number of additional vector entries that should exist.

6.57 src/program-execute.cpp File Reference

Define the Tang::Program::execute method.

```
#include <bit>
#include <cmath>
#include "program.hpp"
#include "opcode.hpp"
#include "computedExpressionError.hpp"
#include "computedExpressionInteger.hpp"
#include "computedExpressionFloat.hpp"
#include "computedExpressionBoolean.hpp"
#include "computedExpressionString.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.

6.57.1 Detailed Description

Define the Tang::Program::execute method.

6.57.2 Macro Definition Documentation

6.57.2.1 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.

6.57.2.2 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

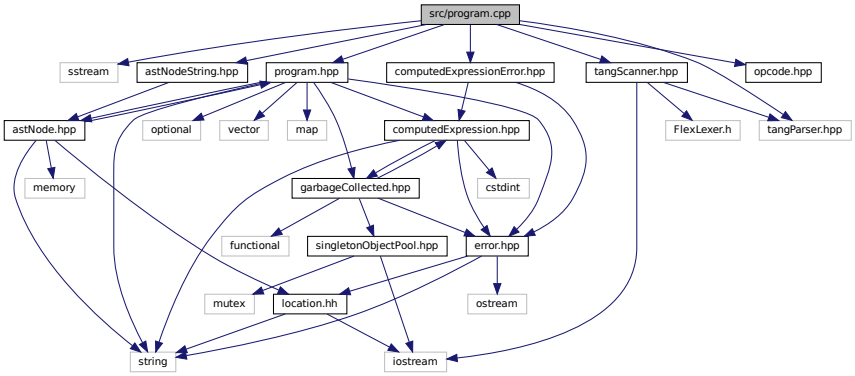
x	The number of entries that should exist in the stack.

6.58 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 "astNodeString.hpp"
#include "computedExpressionError.hpp"
```

Include dependency graph for program.cpp:



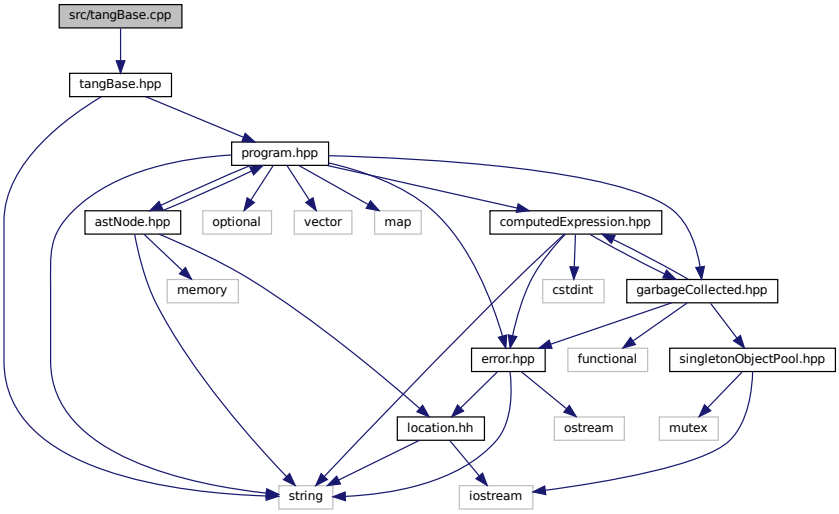
6.58.1 Detailed Description

Define the Tang::Program class.

6.59 src/tangBase.cpp File Reference

Define the Tang::TangBase class.

```
#include "tangBase.hpp"
```
Include dependency graph for tangBase.cpp:



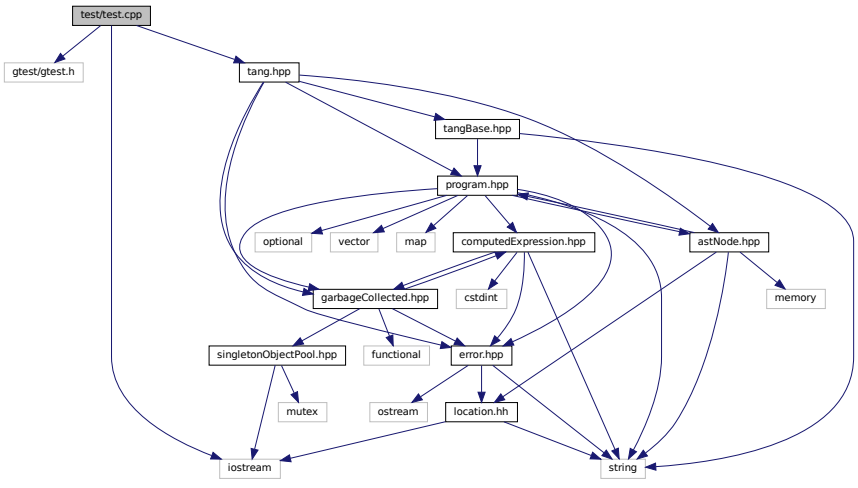
6.59.1 Detailed Description

Define the Tang::TangBase class.

6.60 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** (Declare, Boolean)
- **TEST** (Declare, String)
- **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, Not)
- **TEST** (Expression, LessThan)
- **TEST** (Expression, LessThanEqual)
- **TEST** (Expression, GreaterThan)
- **TEST** (Expression, GreaterThanEqual)
- **TEST** (Expression, Equal)
- **TEST** (Expression, NotEqual)
- **TEST** (Expression, And)
- **TEST** (Expression, Or)
- **TEST** (CodeBlock, Statements)
- **TEST** (Assign, Identifier)
- **TEST** (ControlFlow, IfElse)
- **TEST** (ControlFlow, While)
- **TEST** (ControlFlow, DoWhile)
- **TEST** (ControlFlow, For)
- **TEST** (Print, Default)
- int **main** (int argc, char $**$argv)

6.60.1 Detailed Description

Test the general language behaviors.

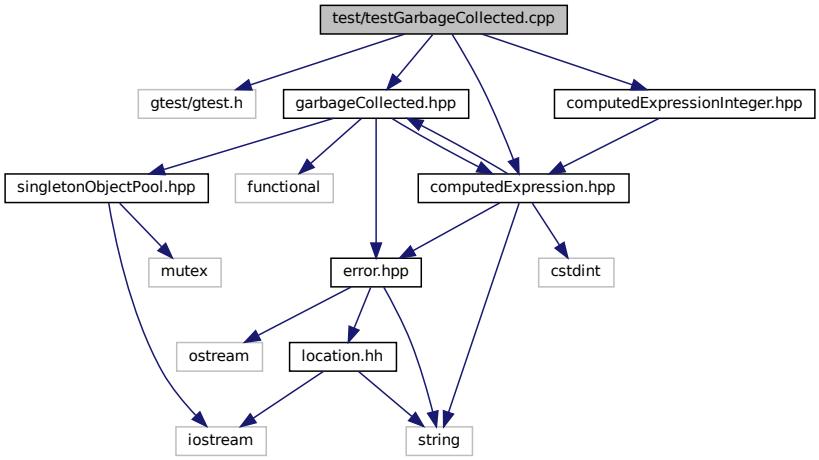
6.61 test/testGarbageCollected.cpp File Reference

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)

6.61.1 Detailed Description

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

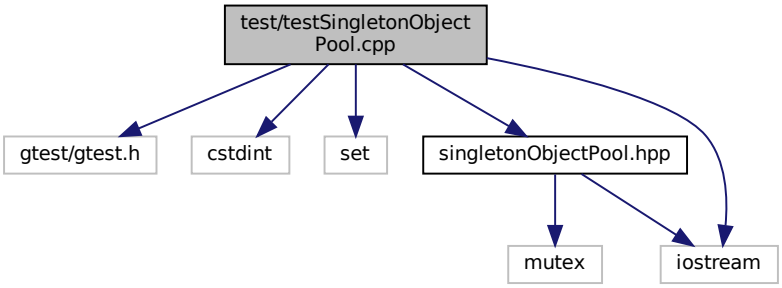
6.62 test/testSingletonObjectPool.cpp File Reference

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)

6.62.1 Detailed Description

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

Index

__add
    Tang::ComputedExpression, 70
    Tang::ComputedExpressionBoolean, 80
    Tang::ComputedExpressionError, 89
    Tang::ComputedExpressionFloat, 100
    Tang::ComputedExpressionInteger, 110
    Tang::ComputedExpressionString, 120
__boolean
    Tang::ComputedExpression, 71
    Tang::ComputedExpressionBoolean, 80
    Tang::ComputedExpressionError, 90
    Tang::ComputedExpressionFloat, 100
    Tang::ComputedExpressionInteger, 110
    Tang::ComputedExpressionString, 120
__divide
    Tang::ComputedExpression, 71
    Tang::ComputedExpressionBoolean, 80
    Tang::ComputedExpressionError, 90
    Tang::ComputedExpressionFloat, 100
    Tang::ComputedExpressionInteger, 110
    Tang::ComputedExpressionString, 120
__equal
    Tang::ComputedExpression, 71
    Tang::ComputedExpressionBoolean, 81
    Tang::ComputedExpressionError, 90
    Tang::ComputedExpressionFloat, 101
    Tang::ComputedExpressionInteger, 111
    Tang::ComputedExpressionString, 121
__float
    Tang::ComputedExpression, 72
    Tang::ComputedExpressionBoolean, 81
    Tang::ComputedExpressionError, 91
    Tang::ComputedExpressionFloat, 101
    Tang::ComputedExpressionInteger, 111
    Tang::ComputedExpressionString, 121
__integer
    Tang::ComputedExpression, 72
    Tang::ComputedExpressionBoolean, 81
    Tang::ComputedExpressionError, 91
    Tang::ComputedExpressionFloat, 101
    Tang::ComputedExpressionInteger, 111
    Tang::ComputedExpressionString, 121
__lessThan
    Tang::ComputedExpression, 72
    Tang::ComputedExpressionBoolean, 82
    Tang::ComputedExpressionError, 91
    Tang::ComputedExpressionFloat, 102
    Tang::ComputedExpressionInteger, 112
    Tang::ComputedExpressionString, 121
__modulo
    Tang::ComputedExpression, 73
    Tang::ComputedExpressionBoolean, 82
    Tang::ComputedExpressionError, 92
    Tang::ComputedExpressionFloat, 102
    Tang::ComputedExpressionInteger, 112
    Tang::ComputedExpressionString, 122
__multiply
    Tang::ComputedExpression, 73
    Tang::ComputedExpressionBoolean, 83
    Tang::ComputedExpressionError, 92
    Tang::ComputedExpressionFloat, 102
    Tang::ComputedExpressionInteger, 112
    Tang::ComputedExpressionString, 122
__negative
    Tang::ComputedExpression, 73
    Tang::ComputedExpressionBoolean, 83
    Tang::ComputedExpressionError, 92
    Tang::ComputedExpressionFloat, 103
    Tang::ComputedExpressionInteger, 113
    Tang::ComputedExpressionString, 123
__not
    Tang::ComputedExpression, 74
    Tang::ComputedExpressionBoolean, 83
    Tang::ComputedExpressionError, 93
    Tang::ComputedExpressionFloat, 103
    Tang::ComputedExpressionInteger, 113
    Tang::ComputedExpressionString, 123
__string
    Tang::ComputedExpression, 74
    Tang::ComputedExpressionBoolean, 83
    Tang::ComputedExpressionError, 93
    Tang::ComputedExpressionFloat, 103
    Tang::ComputedExpressionInteger, 113
    Tang::ComputedExpressionString, 123
__subtract
    Tang::ComputedExpression, 74
    Tang::ComputedExpressionBoolean, 84
    Tang::ComputedExpressionError, 93
    Tang::ComputedExpressionFloat, 104
    Tang::ComputedExpressionInteger, 114
    Tang::ComputedExpressionString, 123
~GarbageCollected
    Tang::GarbageCollected, 132

ADD
    opcode.hpp, 186
Add
    Tang::AstNodeBinary, 20
addBytecode

Tang::Program, 150
And
Tang::AstNodeBinary, 20
AstNode
Tang::AstNode, 13
AstNodeAssign
Tang::AstNodeAssign, 16
AstNodeBinary
Tang::AstNodeBinary, 20
AstNodeBlock
Tang::AstNodeBlock, 24
AstNodeBoolean
Tang::AstNodeBoolean, 27
AstNodeCast
Tang::AstNodeCast, 31
AstNodeDoWhile
Tang::AstNodeDoWhile, 35
AstNodeFloat
Tang::AstNodeFloat, 38
AstNodeFor
Tang::AstNodeFor, 41
AstNodeIdentifier
Tang::AstNodeIdentifier, 45
AstNodeIfElse
Tang::AstNodeIfElse, 48
AstNodeInteger
Tang::AstNodeInteger, 52
AstNodePrint
Tang::AstNodePrint, 55
AstNodeString
Tang::AstNodeString, 59
AstNodeUnary
Tang::AstNodeUnary, 63
AstNodeWhile
Tang::AstNodeWhile, 67

BOOLEAN
opcode.hpp, 186
Boolean
Tang::AstNodeCast, 31
build/generated/location.hh, 159

CASTBOOLEAN
opcode.hpp, 186
CASTFLOAT
opcode.hpp, 186
CASTINTEGER
opcode.hpp, 186
CodeType
Tang::Program, 149
collectIdentifiers
Tang::AstNode, 13
Tang::AstNodeAssign, 17
Tang::AstNodeBinary, 21
Tang::AstNodeBlock, 24
Tang::AstNodeBoolean, 27
Tang::AstNodeCast, 32
Tang::AstNodeDoWhile, 35
Tang::AstNodeFloat, 38
Tang::AstNodeFor, 42
Tang::AstNodeIdentifier, 45
Tang::AstNodeIfElse, 49
Tang::AstNodeInteger, 52
Tang::AstNodePrint, 56
Tang::AstNodeString, 59
Tang::AstNodeUnary, 63
Tang::AstNodeWhile, 67
collectStrings
Tang::AstNode, 14
Tang::AstNodeAssign, 17
Tang::AstNodeBinary, 21
Tang::AstNodeBlock, 24
Tang::AstNodeBoolean, 27
Tang::AstNodeCast, 32
Tang::AstNodeDoWhile, 35
Tang::AstNodeFloat, 39
Tang::AstNodeFor, 42
Tang::AstNodeIdentifier, 45
Tang::AstNodeIfElse, 49
Tang::AstNodeInteger, 52
Tang::AstNodePrint, 56
Tang::AstNodeString, 59
Tang::AstNodeUnary, 64
Tang::AstNodeWhile, 67
compile
Tang::AstNode, 14
Tang::AstNodeAssign, 17
Tang::AstNodeBinary, 21
Tang::AstNodeBlock, 25
Tang::AstNodeBoolean, 29
Tang::AstNodeCast, 32
Tang::AstNodeDoWhile, 36
Tang::AstNodeFloat, 39
Tang::AstNodeFor, 42
Tang::AstNodeIdentifier, 46
Tang::AstNodeIfElse, 49
Tang::AstNodeInteger, 53
Tang::AstNodePrint, 56
Tang::AstNodeString, 60
Tang::AstNodeUnary, 64
Tang::AstNodeWhile, 68
compileLiteral
Tang::AstNodeString, 60
compileScript
Tang::TangBase, 154
ComputedExpressionBoolean
Tang::ComputedExpressionBoolean, 80
ComputedExpressionError
Tang::ComputedExpressionError, 89
ComputedExpressionFloat
Tang::ComputedExpressionFloat, 99
ComputedExpressionInteger
Tang::ComputedExpressionInteger, 109
ComputedExpressionString
Tang::ComputedExpressionString, 119

Default
Tang::AstNodePrint, 55

DIVIDE
opcode.hpp, 186
Divide
Tang::AstNodeBinary, 20
dump
Tang::AstNode, 14
Tang::AstNodeAssign, 18
Tang::AstNodeBinary, 22
Tang::AstNodeBlock, 25
Tang::AstNodeBoolean, 29
Tang::AstNodeCast, 33
Tang::AstNodeDoWhile, 36
Tang::AstNodeFloat, 40
Tang::AstNodeFor, 43
Tang::AstNodeIdentifier, 46
Tang::AstNodeIfElse, 50
Tang::AstNodeInteger, 53
Tang::AstNodePrint, 57
Tang::AstNodeString, 61
Tang::AstNodeUnary, 65
Tang::AstNodeWhile, 68
Tang::ComputedExpression, 75
Tang::ComputedExpressionBoolean, 84
Tang::ComputedExpressionError, 94
Tang::ComputedExpressionFloat, 104
Tang::ComputedExpressionInteger, 114
Tang::ComputedExpressionString, 124
dumpBytecode
Tang::Program, 150
DUMPPROGRAMCHECK
program-dumpBytecode.cpp, 210

EQ
opcode.hpp, 186
Equal
Tang::AstNodeBinary, 20
Error
Tang::Error, 128
error.cpp
operator<<, 208
execute
Tang::Program, 150
EXECUTEPROGRAMCHECK
program-execute.cpp, 211

FLOAT
opcode.hpp, 186
Float
Tang::AstNodeCast, 31

GarbageCollected
Tang::GarbageCollected, 131, 132
get
Tang::SingletonObjectPool< T >, 153
get_next_token
Tang::TangScanner, 156
getAst
Tang::Program, 151
getBytecode
Tang::Program, 151
getCode
Tang::Program, 151
getInstance
Tang::SingletonObjectPool< T >, 153
getResult
Tang::Program, 151
GreaterThan
Tang::AstNodeBinary, 20
GreaterThanEqual
Tang::AstNodeBinary, 20
GT
opcode.hpp, 186
GTE
opcode.hpp, 186

include/astNode.hpp, 161
include/astNodeAssign.hpp, 162
include/astNodeBinary.hpp, 163
include/astNodeBlock.hpp, 164
include/astNodeBoolean.hpp, 165
include/astNodeCast.hpp, 166
include/astNodeDoWhile.hpp, 167
include/astNodeFloat.hpp, 168
include/astNodeFor.hpp, 169
include/astNodeIdentifier.hpp, 170
include/astNodeIfElse.hpp, 171
include/astNodeInteger.hpp, 172
include/astNodePrint.hpp, 173
include/astNodeString.hpp, 174
include/astNodeUnary.hpp, 175
include/astNodeWhile.hpp, 176
include/computedExpression.hpp, 177
include/computedExpressionBoolean.hpp, 178
include/computedExpressionError.hpp, 179
include/computedExpressionFloat.hpp, 180
include/computedExpressionInteger.hpp, 181
include/computedExpressionString.hpp, 182
include/error.hpp, 183
include/garbageCollected.hpp, 184
include/macros.hpp, 184
include/opcode.hpp, 185
include/program.hpp, 187
include/singletonObjectPool.hpp, 188
include/tang.hpp, 189
include/tangBase.hpp, 190
include/tangScanner.hpp, 191
INTEGER
opcode.hpp, 186
Integer
Tang::AstNodeCast, 31
is_equal
Tang::ComputedExpression, 75–77
Tang::ComputedExpressionBoolean, 84–86
Tang::ComputedExpressionError, 94, 96, 97
Tang::ComputedExpressionFloat, 105, 106
Tang::ComputedExpressionInteger, 115, 116
Tang::ComputedExpressionString, 124–126

JMP
opcode.hpp, 186
JMPF
opcode.hpp, 186
JMPF_POP
opcode.hpp, 186
JMPT
opcode.hpp, 186
JMPT_POP
opcode.hpp, 186

LessThan
Tang::AstNodeBinary, 20
LessThanEqual
Tang::AstNodeBinary, 20
location.hh
operator<<, 160, 161
LT
opcode.hpp, 186
LTE
opcode.hpp, 186

macros.hpp
TANG_UNUSED, 185
make
Tang::GarbageCollected, 132
makeCopy
Tang::ComputedExpression, 77
Tang::ComputedExpressionBoolean, 87
Tang::ComputedExpressionError, 97
Tang::ComputedExpressionFloat, 107
Tang::ComputedExpressionInteger, 117
Tang::ComputedExpressionString, 126
MODULO
opcode.hpp, 186
Modulo
Tang::AstNodeBinary, 20
MULTIPLY
opcode.hpp, 186
Multiply
Tang::AstNodeBinary, 20

NEGATIVE
opcode.hpp, 186
Negative
Tang::AstNodeUnary, 63
NEQ
opcode.hpp, 186
NOT
opcode.hpp, 186
Not
Tang::AstNodeUnary, 63
NotEqual
Tang::AstNodeBinary, 20
NULLVAL
opcode.hpp, 186

Opcode
opcode.hpp, 186
opcode.hpp
ADD, 186
BOOLEAN, 186
CASTBOOLEAN, 186
CASTFLOAT, 186
CASTINTEGER, 186
DIVIDE, 186
EQ, 186
FLOAT, 186
GT, 186
GTE, 186
INTEGER, 186
JMP, 186
JMPF, 186
JMPF_POP, 186
JMPT, 186
JMPT_POP, 186
LT, 186
LTE, 186
MODULO, 186
MULTIPLY, 186
NEGATIVE, 186
NEQ, 186
NOT, 186
NULLVAL, 186
Opcode, 186
PEEK, 186
POKE, 186
POP, 186
PRINT, 186
STRING, 186
SUBTRACT, 186
Operation
Tang::AstNodeBinary, 20
Operator
Tang::AstNodeUnary, 63
operator!
Tang::GarbageCollected, 133
operator!=
Tang::GarbageCollected, 133
operator<
Tang::GarbageCollected, 138
operator<<
error.cpp, 208
location.hh, 160, 161
Tang::Error, 128
Tang::GarbageCollected, 144
operator<=
Tang::GarbageCollected, 138
operator>
Tang::GarbageCollected, 143
operator>=
Tang::GarbageCollected, 143
operator*
Tang::GarbageCollected, 134, 135
operator+
Tang::GarbageCollected, 135
operator-

Tang::GarbageCollected, 136
operator->
Tang::GarbageCollected, 137
operator/
Tang::GarbageCollected, 137
operator=
Tang::GarbageCollected, 139
operator==
Tang::GarbageCollected, 140–142
operator%
Tang::GarbageCollected, 134
Or
Tang::AstNodeBinary, 20

PEEK
opcode.hpp, 186
POKE
opcode.hpp, 186
POP
opcode.hpp, 186
PRINT
opcode.hpp, 186
Program
Tang::Program, 149
program-dumpBytecode.cpp
DUMPPROGRAMCHECK, 210
program-execute.cpp
EXECUTEPROGRAMCHECK, 211
STACKCHECK, 211

recycle
Tang::SingletonObjectPool< T >, 153

Script
Tang::Program, 149
setJumpTarget
Tang::Program, 152
src/astNode.cpp, 192
src/astNodeAssign.cpp, 192
src/astNodeBinary.cpp, 193
src/astNodeBlock.cpp, 194
src/astNodeBoolean.cpp, 194
src/astNodeCast.cpp, 195
src/astNodeDoWhile.cpp, 196
src/astNodeFloat.cpp, 197
src/astNodeFor.cpp, 198
src/astNodeIdentifier.cpp, 198
src/astNodeIfElse.cpp, 199
src/astNodeInteger.cpp, 200
src/astNodePrint.cpp, 201
src/astNodeString.cpp, 201
src/astNodeUnary.cpp, 202
src/astNodeWhile.cpp, 203
src/computedExpression.cpp, 204
src/computedExpressionBoolean.cpp, 204
src/computedExpressionError.cpp, 205
src/computedExpressionFloat.cpp, 206
src/computedExpressionInteger.cpp, 206
src/computedExpressionString.cpp, 207
src/error.cpp, 208
src/program-dumpBytecode.cpp, 209
src/program-execute.cpp, 210
src/program.cpp, 212
src/tangBase.cpp, 212
STACKCHECK
program-execute.cpp, 211
STRING
opcode.hpp, 186
SUBTRACT
opcode.hpp, 186
Subtract
Tang::AstNodeBinary, 20

Tang::AstNode, 11
AstNode, 13
collectIdentifiers, 13
collectStrings, 14
compile, 14
dump, 14
Tang::AstNodeAssign, 15
AstNodeAssign, 16
collectIdentifiers, 17
collectStrings, 17
compile, 17
dump, 18
Tang::AstNodeBinary, 18
Add, 20
And, 20
AstNodeBinary, 20
collectIdentifiers, 21
collectStrings, 21
compile, 21
Divide, 20
dump, 22
Equal, 20
GreaterThan, 20
GreaterThanEqual, 20
LessThan, 20
LessThanEqual, 20
Modulo, 20
Multiply, 20
NotEqual, 20
Operation, 20
Or, 20
Subtract, 20
Tang::AstNodeBlock, 22
AstNodeBlock, 24
collectIdentifiers, 24
collectStrings, 24
compile, 25
dump, 25
Tang::AstNodeBoolean, 26
AstNodeBoolean, 27
collectIdentifiers, 27
collectStrings, 27
compile, 29
dump, 29
Tang::AstNodeCast, 30

AstNodeCast, 31
Boolean, 31
collectIdentifiers, 32
collectStrings, 32
compile, 32
dump, 33
Float, 31
Integer, 31
Type, 31
Tang::AstNodeDoWhile, 33
AstNodeDoWhile, 35
collectIdentifiers, 35
collectStrings, 35
compile, 36
dump, 36
Tang::AstNodeFloat, 37
AstNodeFloat, 38
collectIdentifiers, 38
collectStrings, 39
compile, 39
dump, 40
Tang::AstNodeFor, 40
AstNodeFor, 41
collectIdentifiers, 42
collectStrings, 42
compile, 42
dump, 43
Tang::AstNodeIdentifier, 44
AstNodeIdentifier, 45
collectIdentifiers, 45
collectStrings, 45
compile, 46
dump, 46
Tang::AstNodeIfElse, 47
AstNodeIfElse, 48
collectIdentifiers, 49
collectStrings, 49
compile, 49
dump, 50
Tang::AstNodeInteger, 51
AstNodeInteger, 52
collectIdentifiers, 52
collectStrings, 52
compile, 53
dump, 53
Tang::AstNodePrint, 54
AstNodePrint, 55
collectIdentifiers, 56
collectStrings, 56
compile, 56
Default, 55
dump, 57
Type, 55
Tang::AstNodeString, 57
AstNodeString, 59
collectIdentifiers, 59
collectStrings, 59
compile, 60
compileLiteral, 60
dump, 61
Tang::AstNodeUnary, 61
AstNodeUnary, 63
collectIdentifiers, 63
collectStrings, 64
compile, 64
dump, 65
Negative, 63
Not, 63
Operator, 63
Tang::AstNodeWhile, 65
AstNodeWhile, 67
collectIdentifiers, 67
collectStrings, 67
compile, 68
dump, 68
Tang::ComputedExpression, 69
__add, 70
__boolean, 71
__divide, 71
__equal, 71
__float, 72
__integer, 72
__lessThan, 72
__modulo, 73
__multiply, 73
__negative, 73
__not, 74
__string, 74
__subtract, 74
dump, 75
is_equal, 75–77
makeCopy, 77
Tang::ComputedExpressionBoolean, 78
__add, 80
__boolean, 80
__divide, 80
__equal, 81
__float, 81
__integer, 81
__lessThan, 82
__modulo, 82
__multiply, 83
__negative, 83
__not, 83
__string, 83
__subtract, 84
ComputedExpressionBoolean, 80
dump, 84
is_equal, 84–86
makeCopy, 87
Tang::ComputedExpressionError, 87
__add, 89
__boolean, 90
__divide, 90
__equal, 90
__float, 91

__integer, 91
__lessThan, 91
__modulo, 92
__multiply, 92
__negative, 92
__not, 93
__string, 93
__subtract, 93
ComputedExpressionError, 89
dump, 94
is_equal, 94, 96, 97
makeCopy, 97
Tang::ComputedExpressionFloat, 98
__add, 100
__boolean, 100
__divide, 100
__equal, 101
__float, 101
__integer, 101
__lessThan, 102
__modulo, 102
__multiply, 102
__negative, 103
__not, 103
__string, 103
__subtract, 104
ComputedExpressionFloat, 99
dump, 104
is_equal, 105, 106
makeCopy, 107
Tang::ComputedExpressionInteger, 107
__add, 110
__boolean, 110
__divide, 110
__equal, 111
__float, 111
__integer, 111
__lessThan, 112
__modulo, 112
__multiply, 112
__negative, 113
__not, 113
__string, 113
__subtract, 114
ComputedExpressionInteger, 109
dump, 114
is_equal, 115, 116
makeCopy, 117
Tang::ComputedExpressionString, 117
__add, 120
__boolean, 120
__divide, 120
__equal, 121
__float, 121
__integer, 121
__lessThan, 121
__modulo, 122
__multiply, 122
__negative, 123
__not, 123
__string, 123
__subtract, 123
ComputedExpressionString, 119
dump, 124
is_equal, 124–126
makeCopy, 126
Tang::Error, 127
Error, 128
operator<<, 128
Tang::GarbageCollected, 129
~GarbageCollected, 132
GarbageCollected, 131, 132
make, 132
operator!, 133
operator!=, 133
operator<, 138
operator<<, 144
operator<=, 138
operator>, 143
operator>=, 143
operator*, 134, 135
operator+, 135
operator-, 136
operator->, 137
operator/, 137
operator=, 139
operator==, 140–142
operator%, 134
Tang::location, 144
Tang::position, 146
Tang::Program, 147
addBytecode, 150
CodeType, 149
dumpBytecode, 150
execute, 150
getAst, 151
getBytecode, 151
getCode, 151
getResult, 151
Program, 149
Script, 149
setJumpTarget, 152
Template, 149
Tang::SingletonObjectPool< T >, 152
get, 153
getInstance, 153
recycle, 153
Tang::TangBase, 154
compileScript, 154
TangBase, 154
Tang::TangScanner, 155
get_next_token, 156
TangScanner, 156
TANG_UNUSED
macros.hpp, 185
TangBase

Tang::TangBase, 154
TangScanner
Tang::TangScanner, 156
Template
Tang::Program, 149
test/test.cpp, 213
test/testGarbageCollected.cpp, 214
test/testSingletonObjectPool.cpp, 215
Type
Tang::AstNodeCast, 31
Tang::AstNodePrint, 55